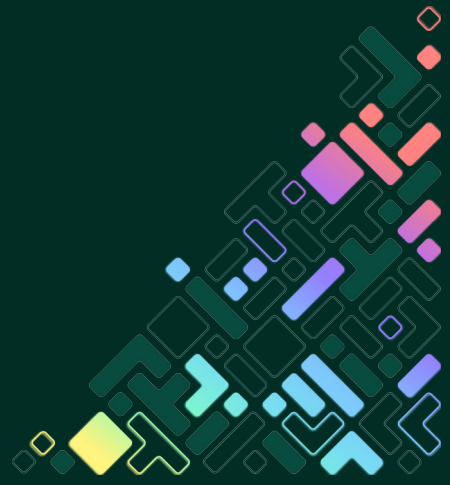
EMBEDDED
LINUX
CONFERENCE

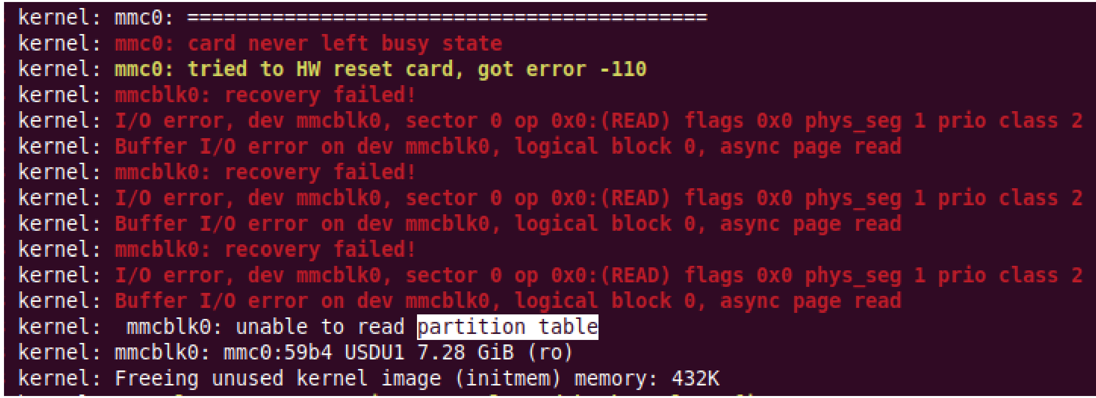
Maximizing SD Card Life, Performance, and Monitoring with KrillKounter

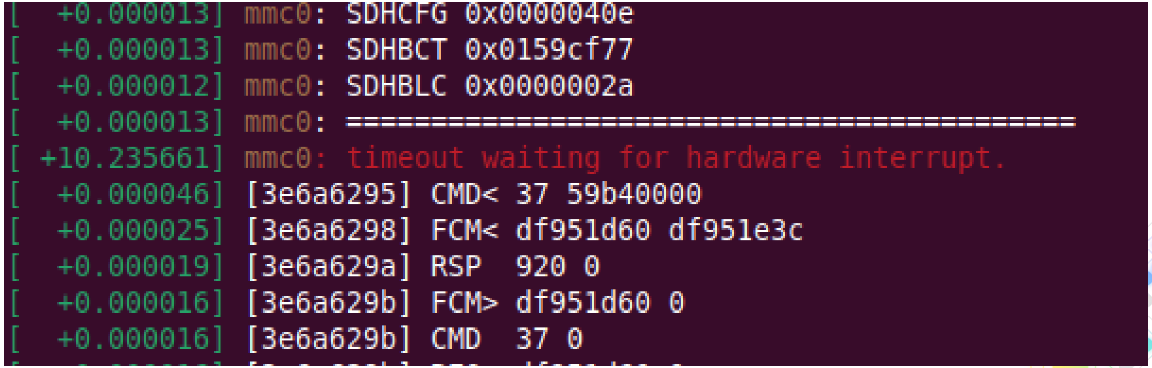
Andrew Murray, The Good Penguin



Typical Issues

- Unreliability
  - Data corruption
  - MMC errors

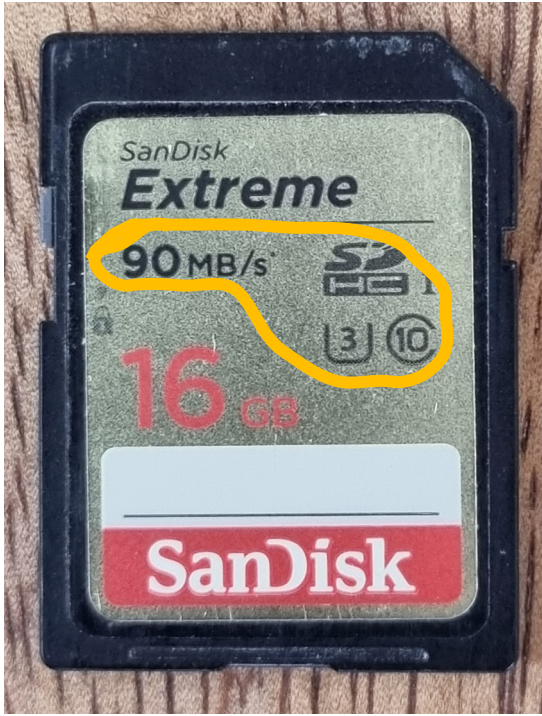




GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Typical Issues

- Poor Performance
  - Not meeting manufacturer claims
- Poor Life
  - Card failures



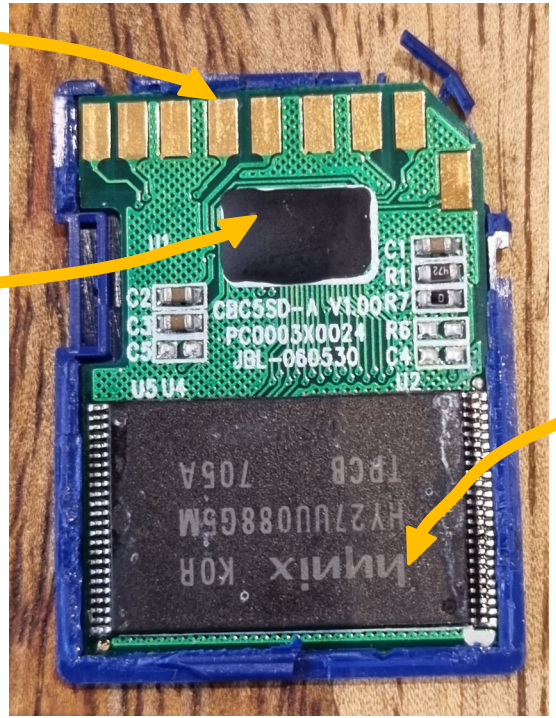
A typical SD Card

2007 era SD card

A typical SD Card

SD Card
Connectors

SD Controller
(microcontroller)



NAND Flash Memory Chip
Hynix 8Gbit SLC

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

NAND is Everywhere

SD / microSD

NAND

eMMC

NAND Inside

SSD

Image: https://www.westerndigital.com/en-se/products/outlet/internal-drives/wd-green-sata-2-5-ssd

NAND Inside

USB

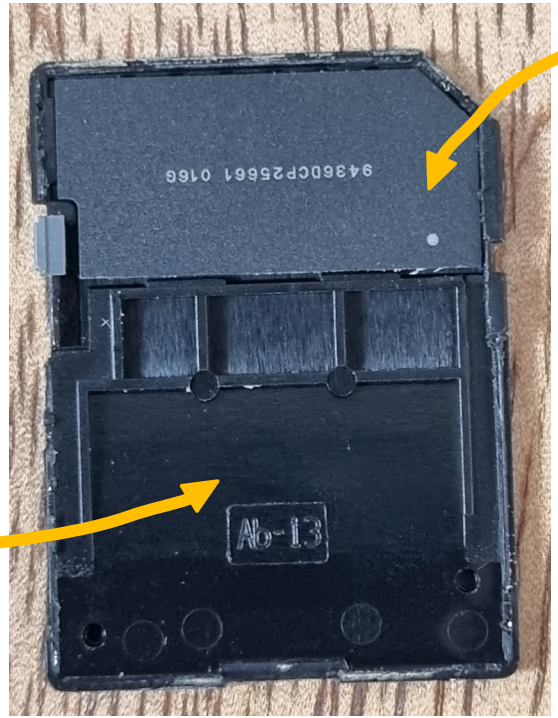
Image: https://commons.wikimedia.org/wiki/File:SanDisk-Cruzer-USB-4GB-ThumbDrive.jpg

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

A typical SD Card

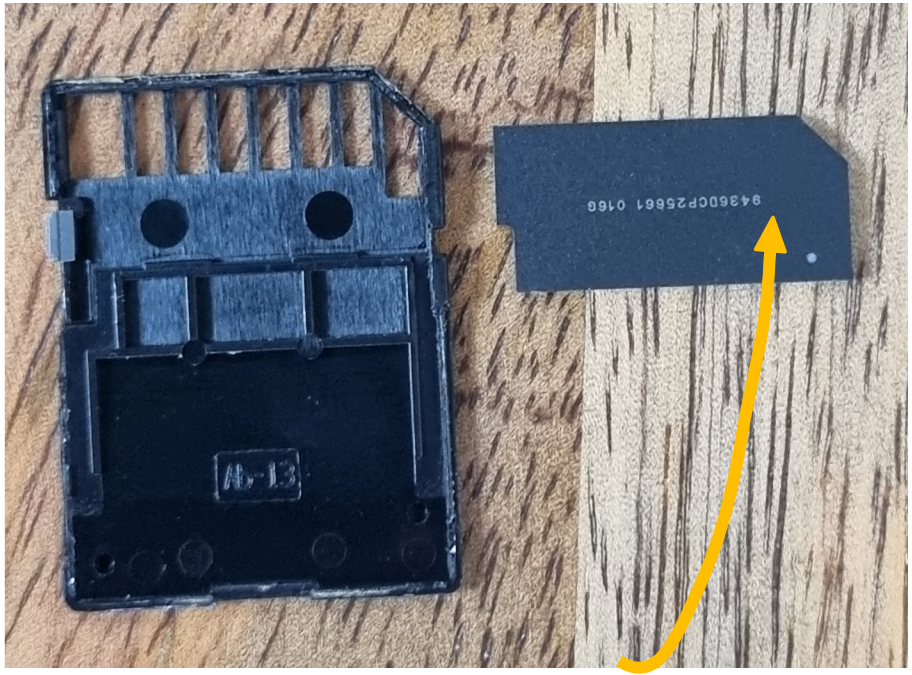
2023 era SD card

A typical SD Card



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

A typical SD Card

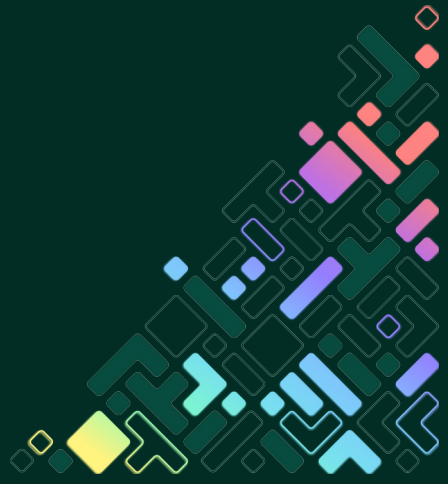


NAND and Controller

SD Card
Connectors

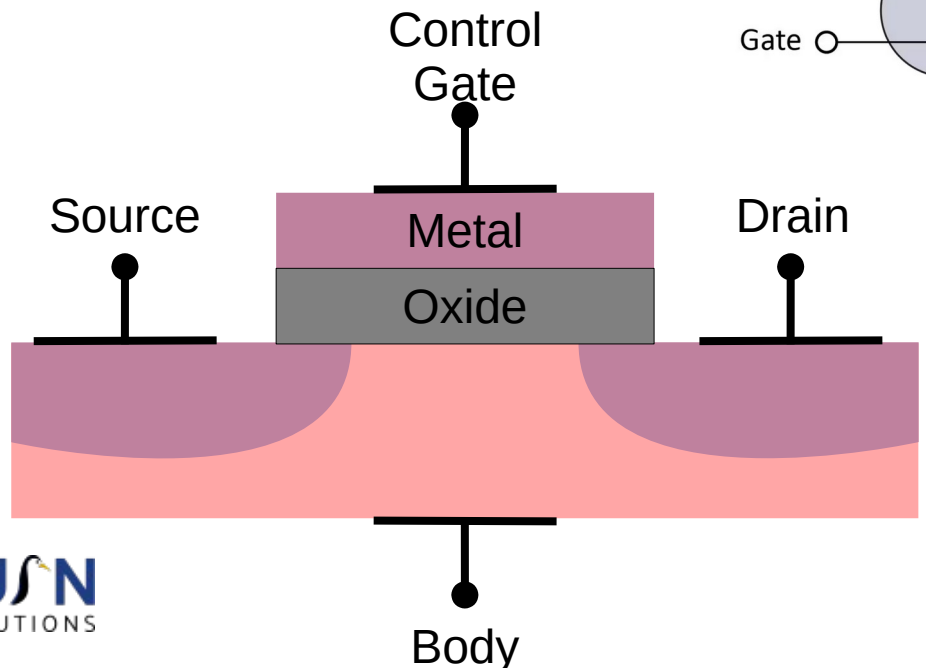
GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

NAND 101



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

MOSFET



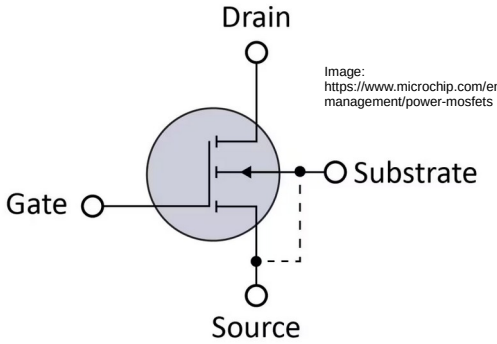


Image:
https://www.microchip.com/en-us/products/power-management/power-mosfets

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

MOSFET

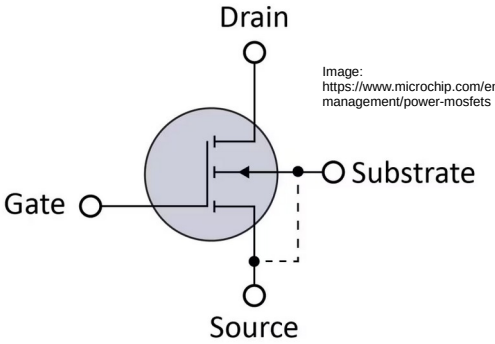
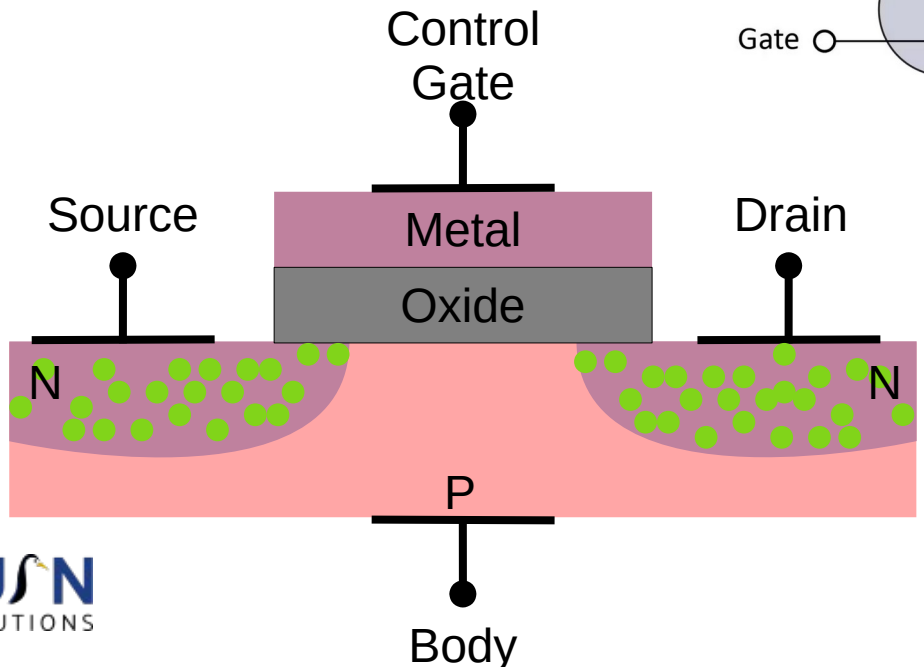


Image: https://www.microchip.com/en-us/products/power-management/power-mosfets



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

MOSFET

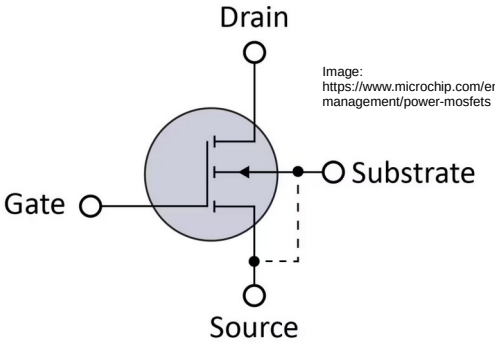
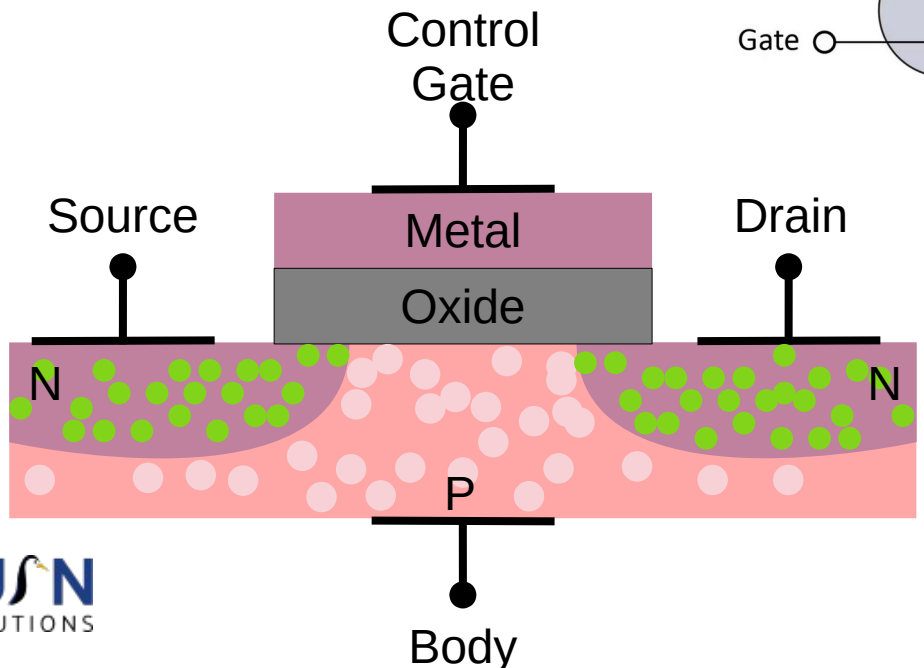


Image: https://www.microchip.com/en-us/products/power-management/power-mosfets



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

MOSFET

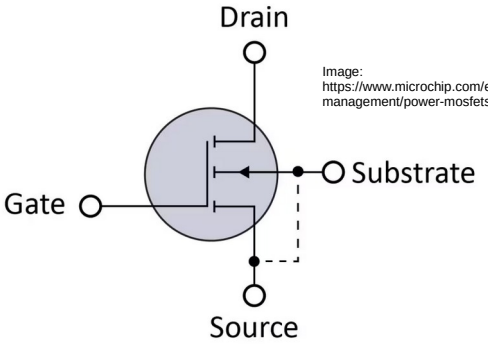
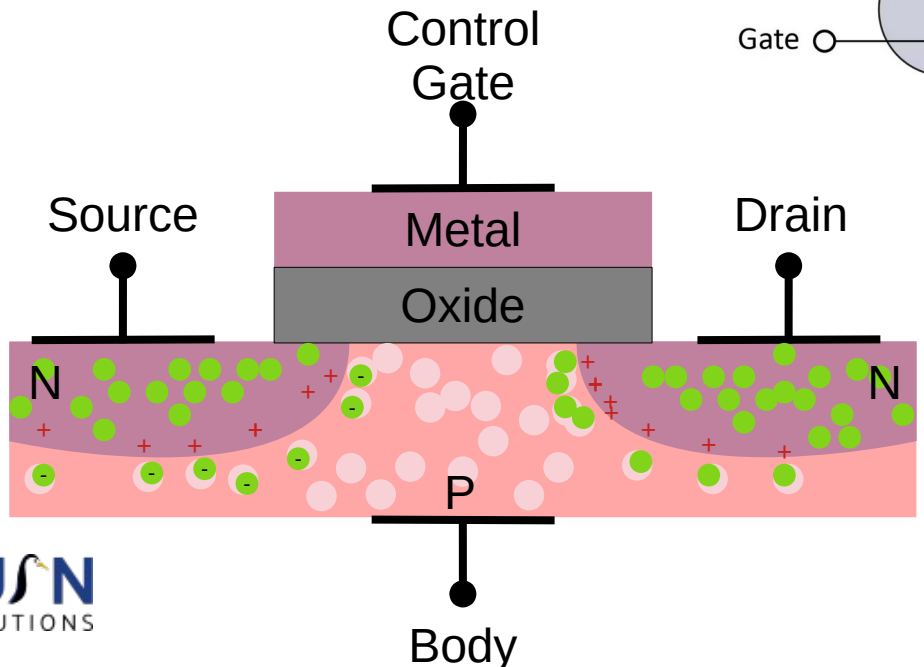


Image: https://www.microchip.com/en-us/products/power-management/power-mosfets



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

MOSFET

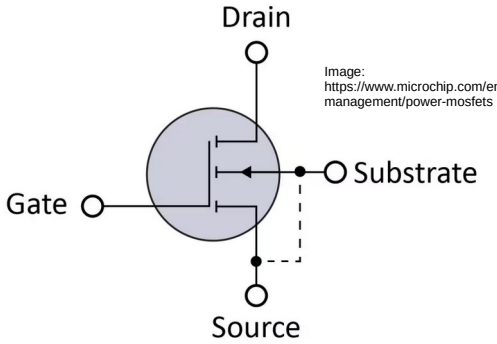
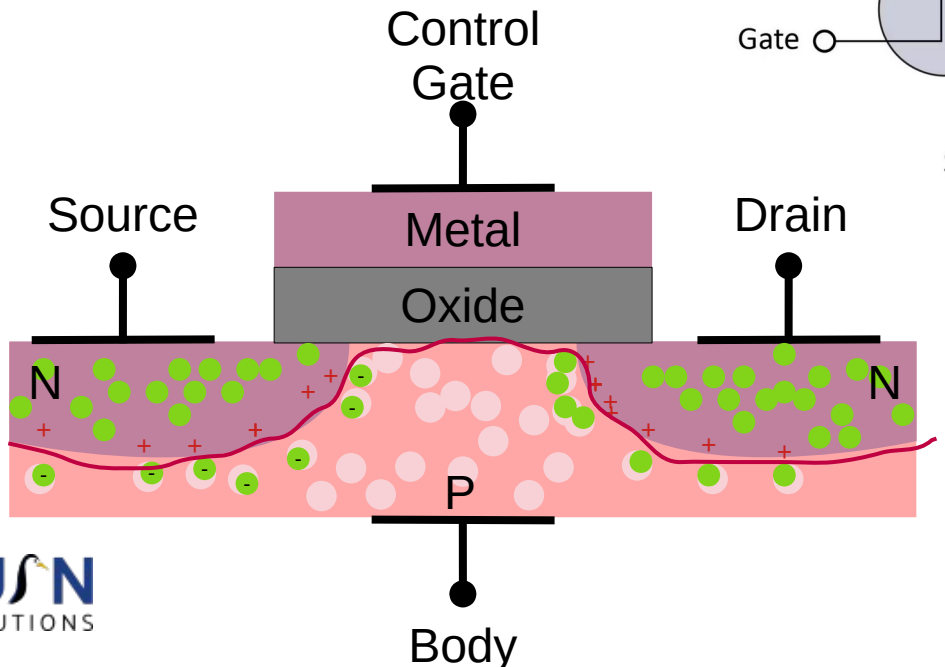
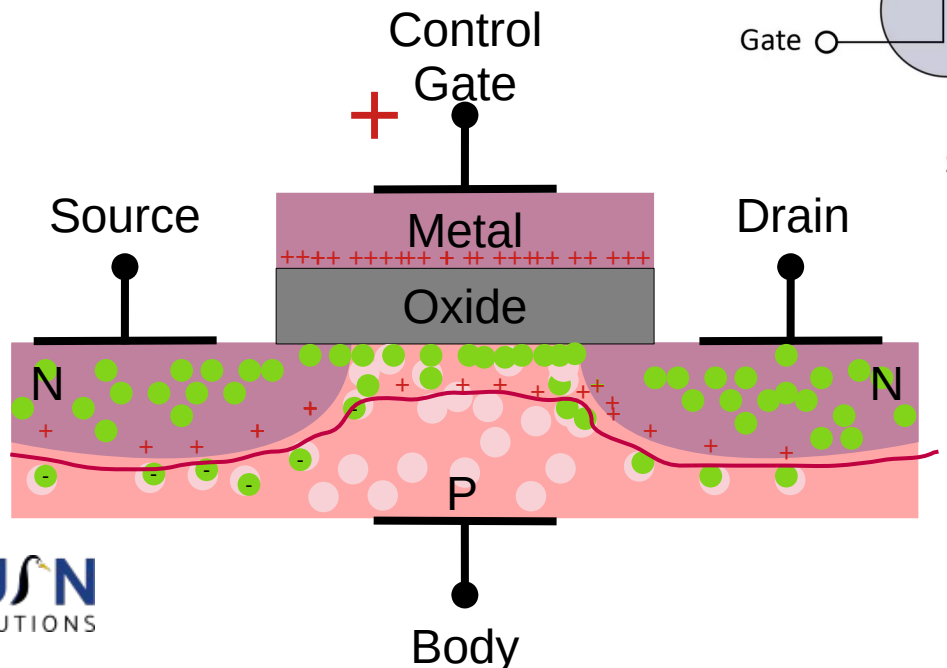


Image: https://www.microchip.com/en-us/products/power-management/power-mosfets



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

MOSFET



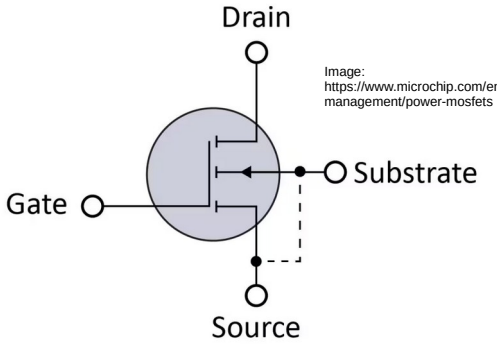
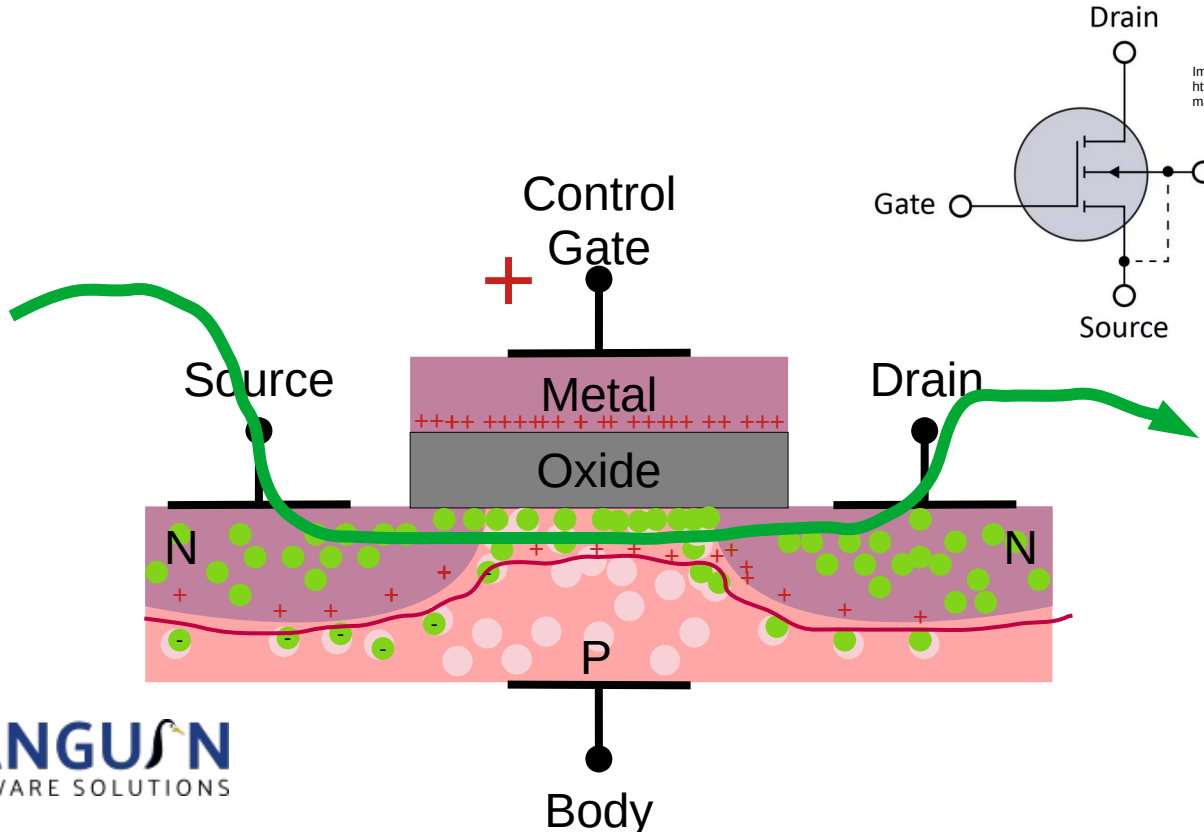


Image:
https://www.microchip.com/en-us/products/power-management/power-mosfets

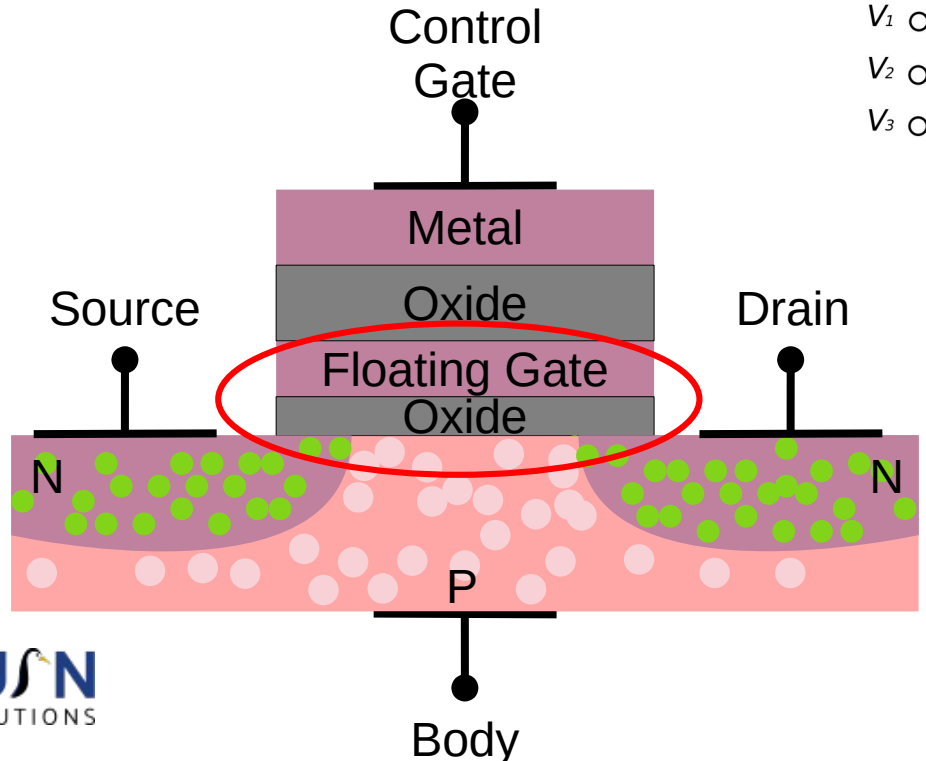
GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

MOSFET



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Floating Gate MOSFET



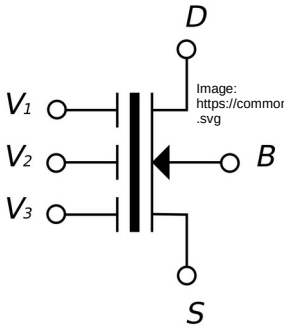
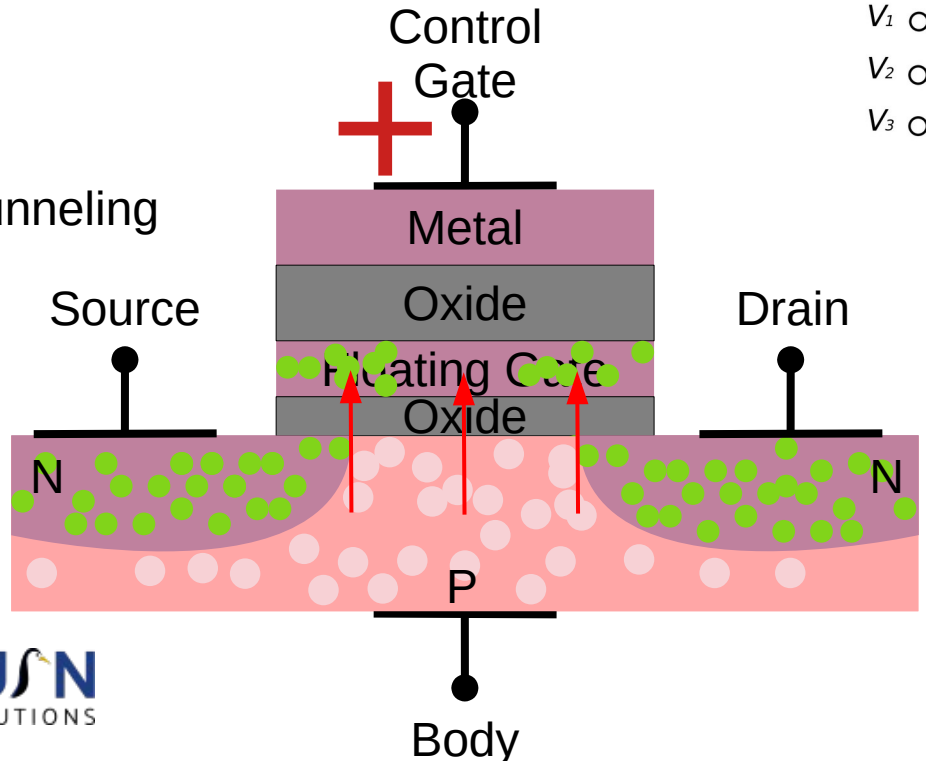


Image: https://commons.wikimedia.org/wiki/File:FGMOS_Symbol.svg

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Floating Gate MOSFET

fowler–nordheim tunneling



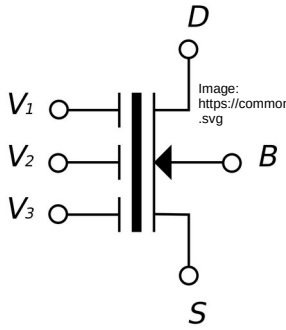
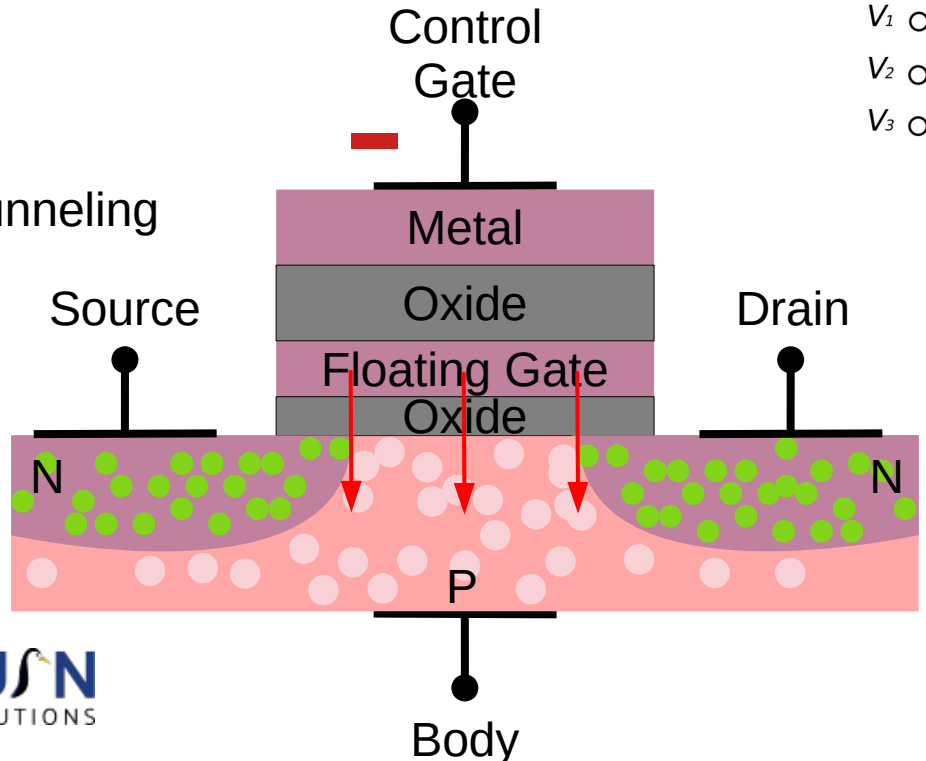


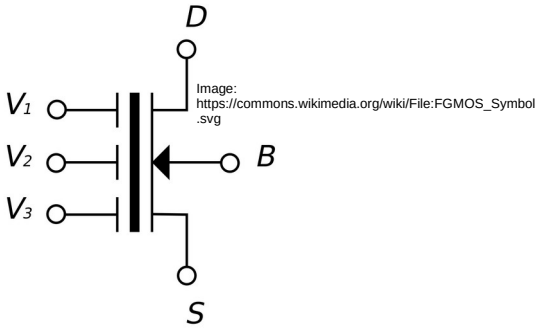
Image: https://commons.wikimedia.org/wiki/File:FGMOS_Symbol.svg

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Floating Gate MOSFET

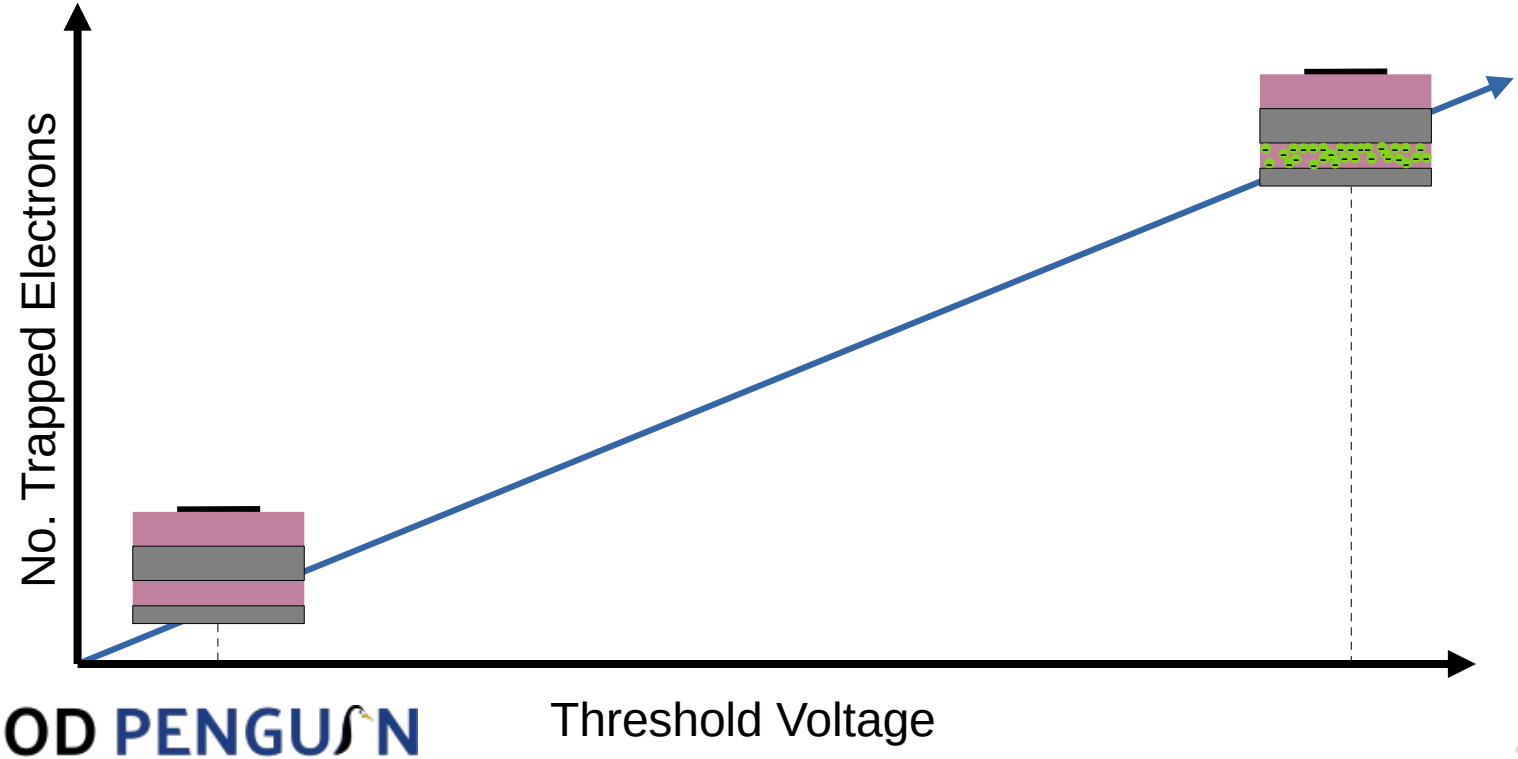
fowler–nordheim tunneling



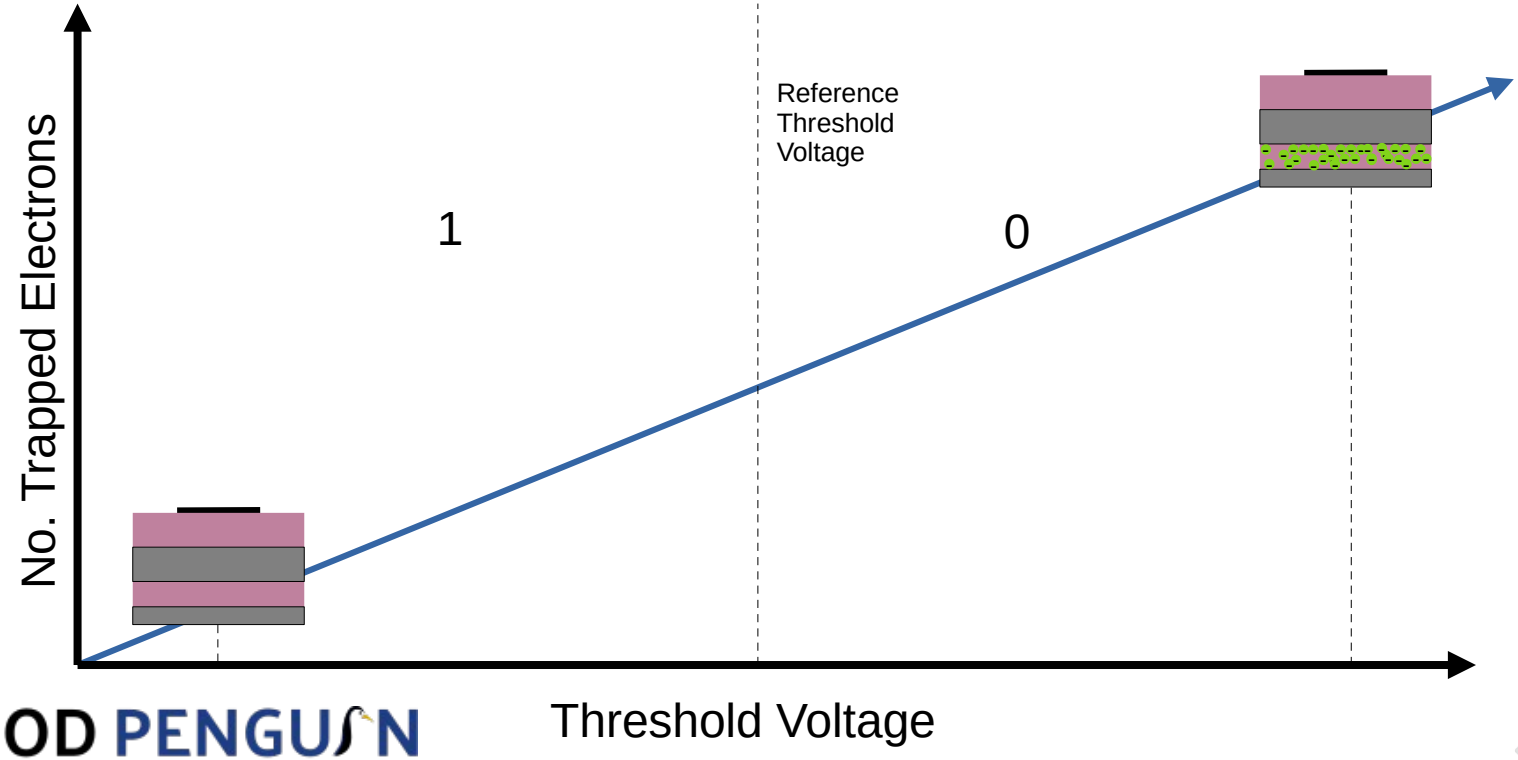


GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Threshold Voltage Varies with Electrons

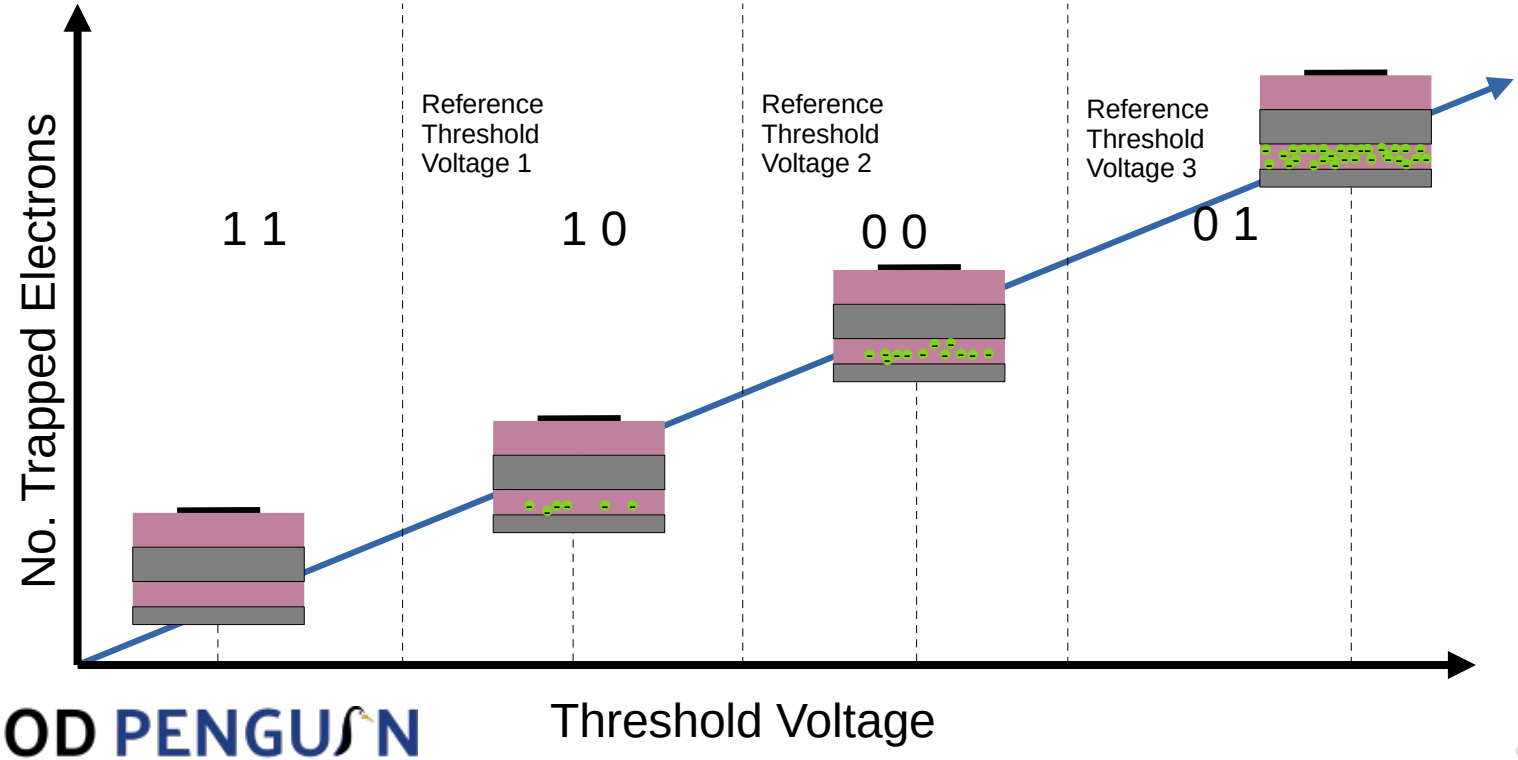


SLC



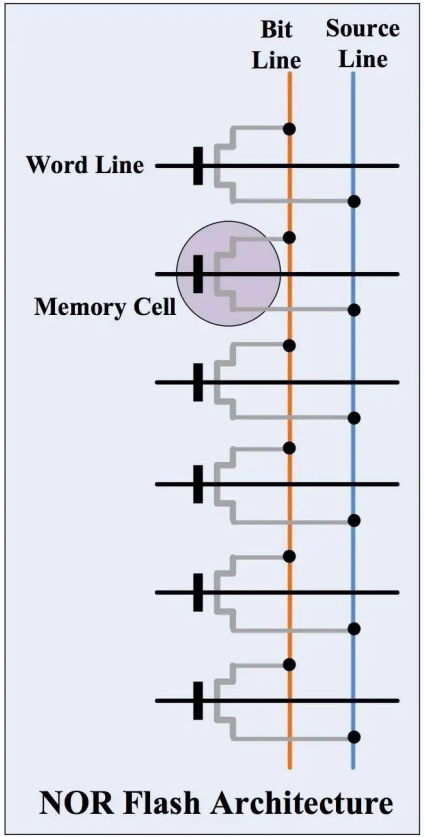
GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

MLC



NAND vs NOR

- NAND/NOR both uses floating gates
- Difference is how they are connected (series/parallel)
- Significant side effect:
  - Erase in blocks
  - Read/Write in pages



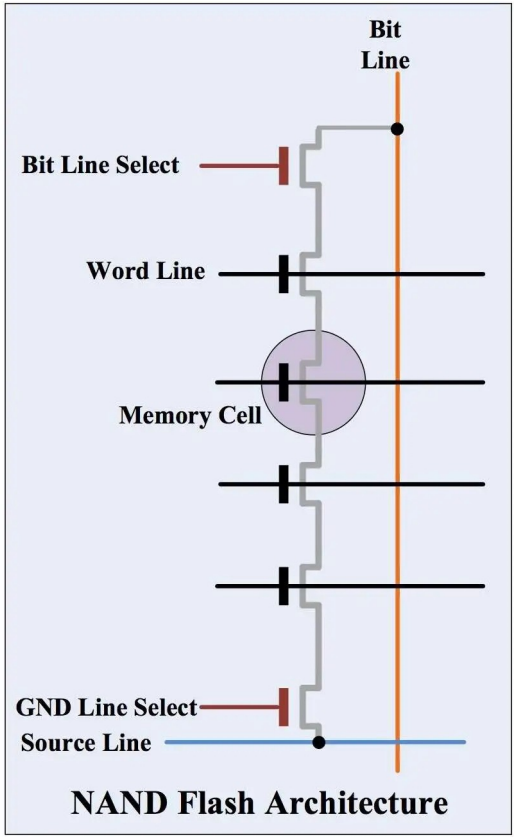
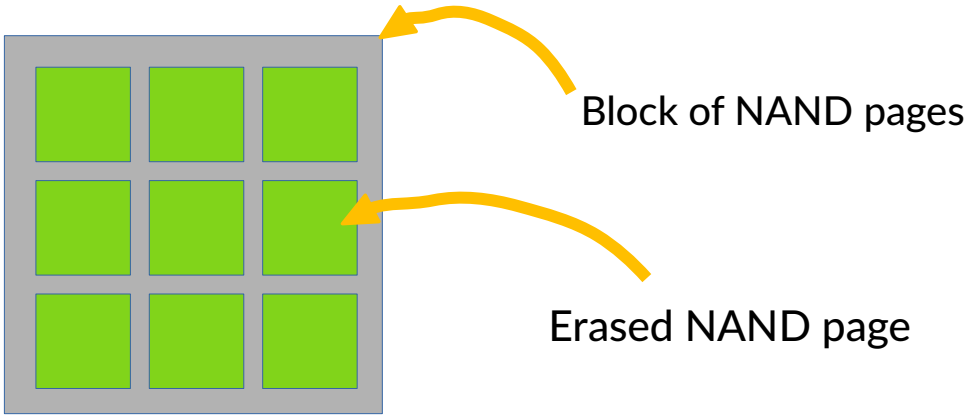


Image: https://www.embedded.com/flash-101-nand-flash-vs-nor-flash/

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

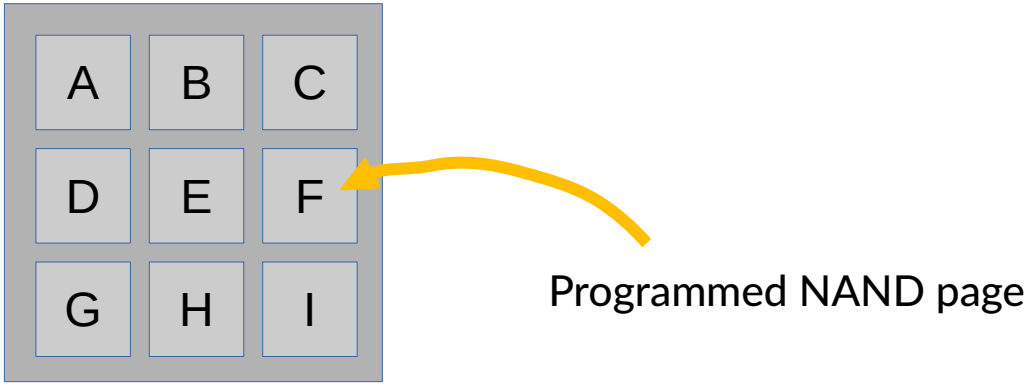
Access Limitations



- We start with a single block of erased NAND

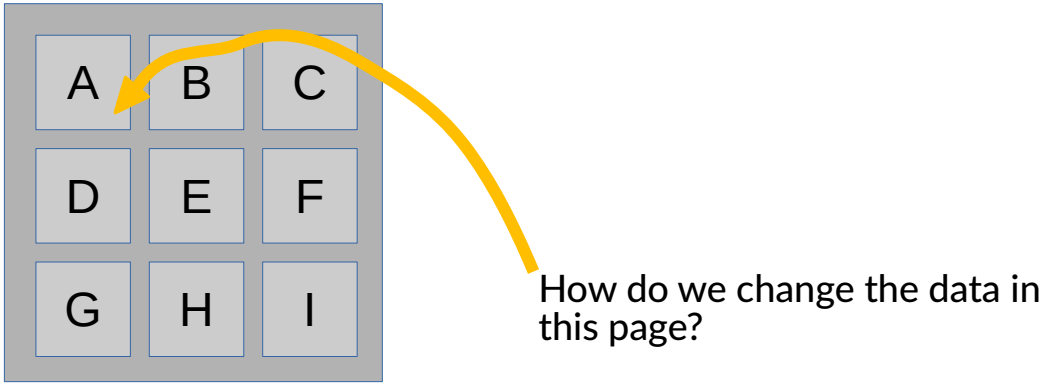
Access Limitations

- Let’s write each page

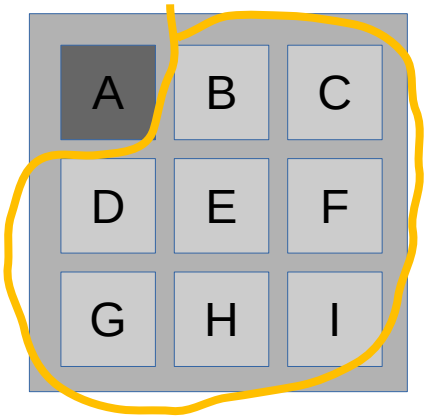


Access Limitations

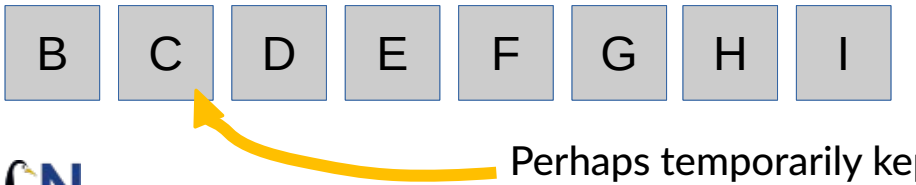
- Let’s write each page



Access Limitations

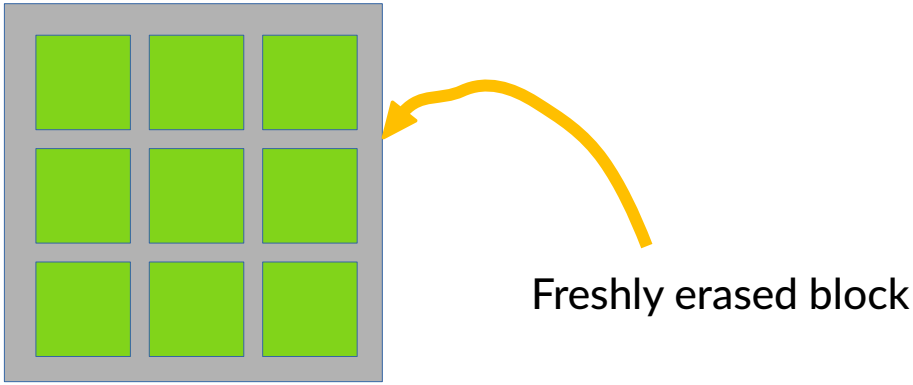


- A simple approach would first copy all the data that we need to keep



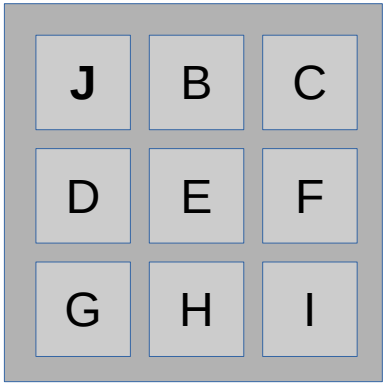
Access Limitations

- Next we erase the whole block



B	C	D	E	F	G	H	I

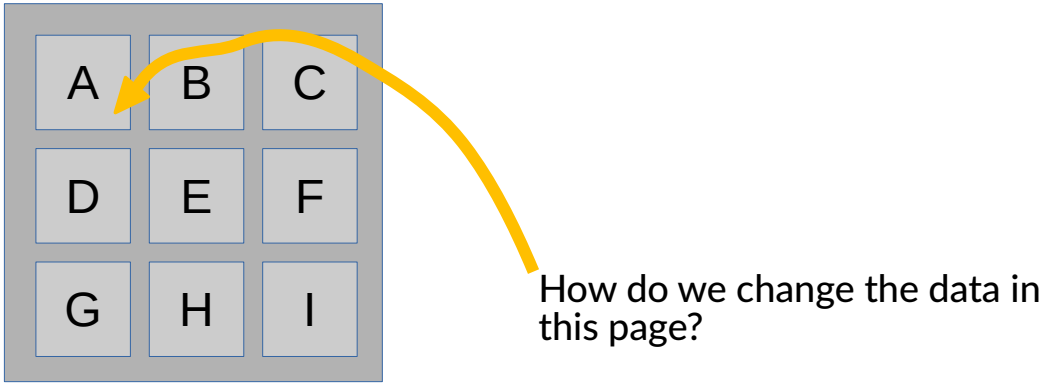
Write Amplification



- Then we write it back
- Host wrote 1 page
- SD Controller wrote 9 pages
- Write Amplification Factor of 9x
- Will be slower

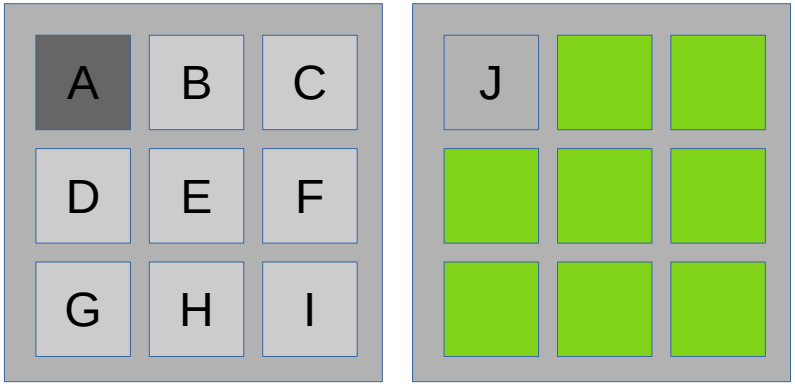
GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Access Limitations



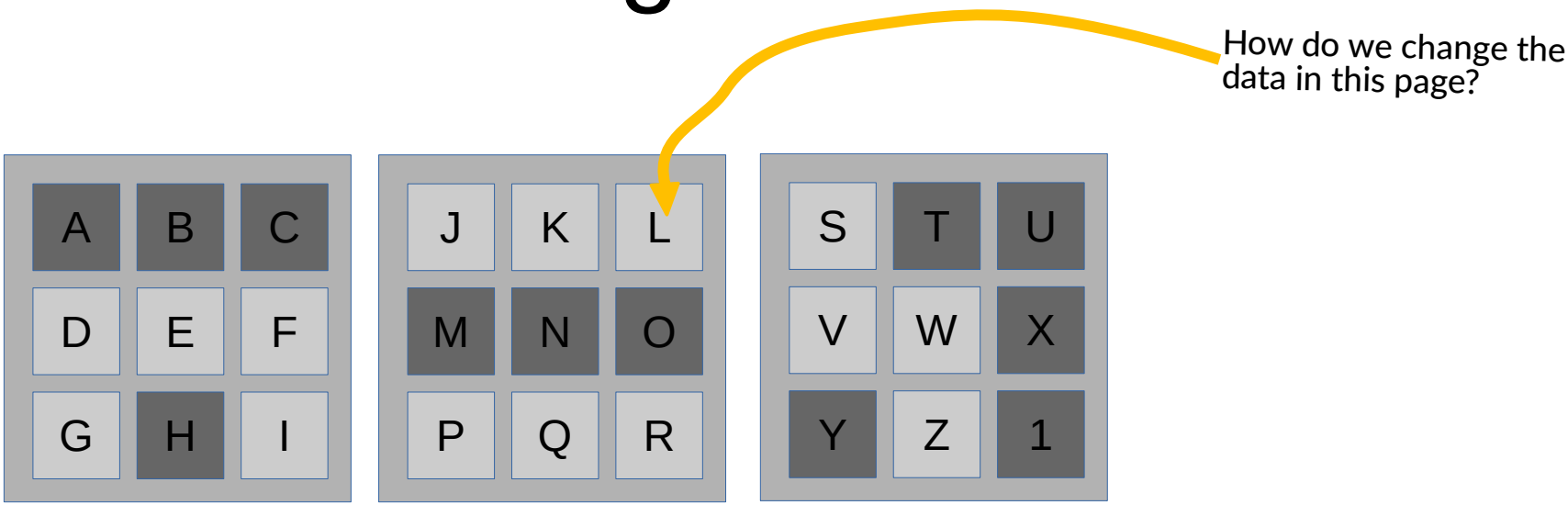
- A more sensible approach

Access Limitations



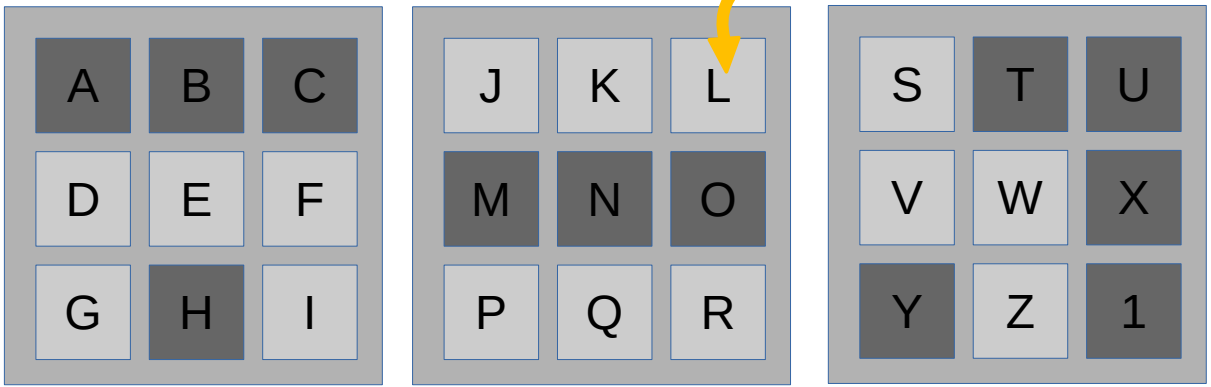
- Instead lets write out of place
- You now have a log based flash translation layer mapping logical blocks to device blocks

Garbage Collection



Garbage Collection

How do we change the data in this page?

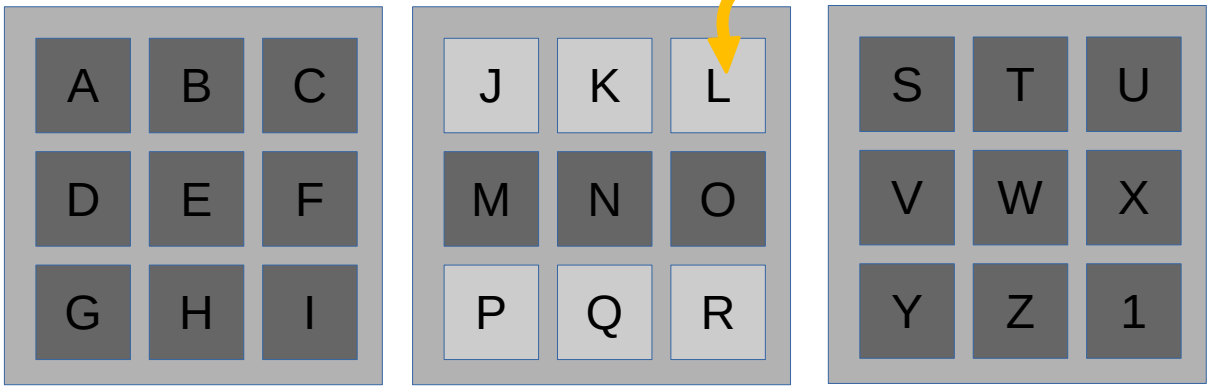


D E F G I S V W Z

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Garbage Collection

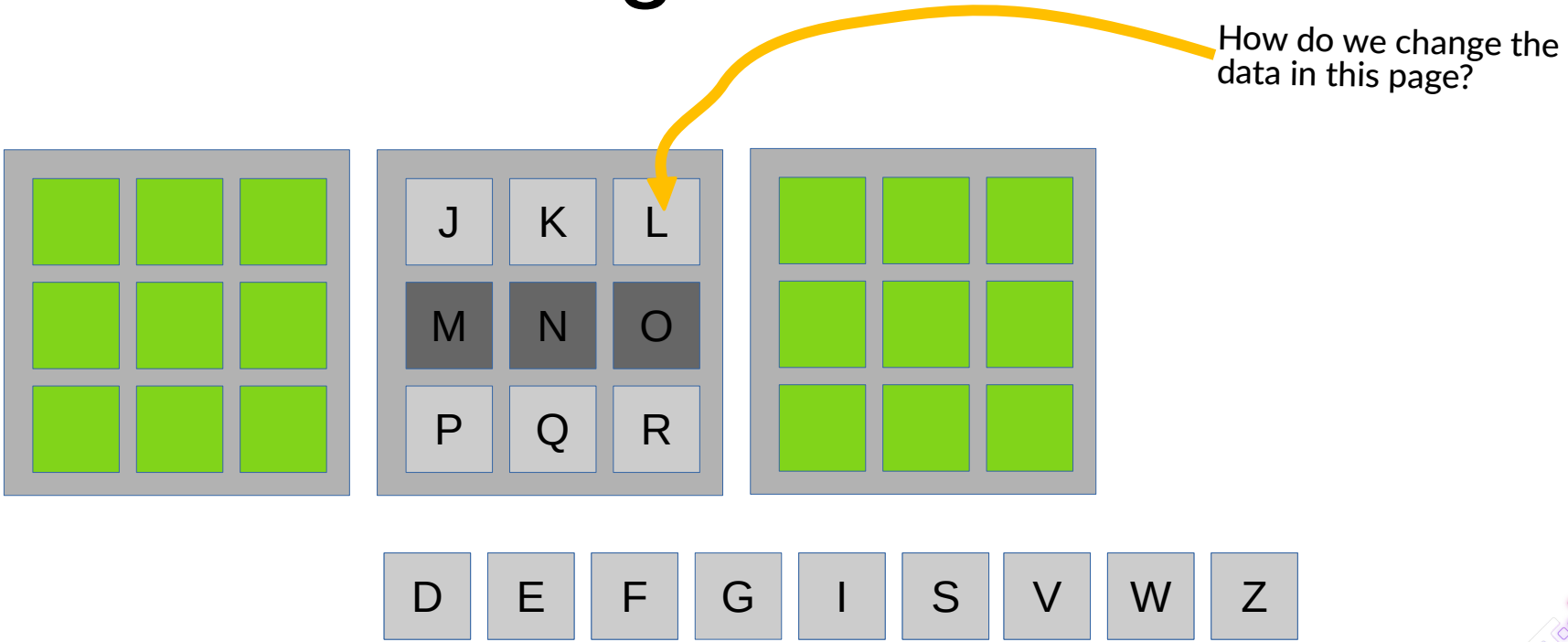
How do we change the data in this page?



D E F G I S V W Z

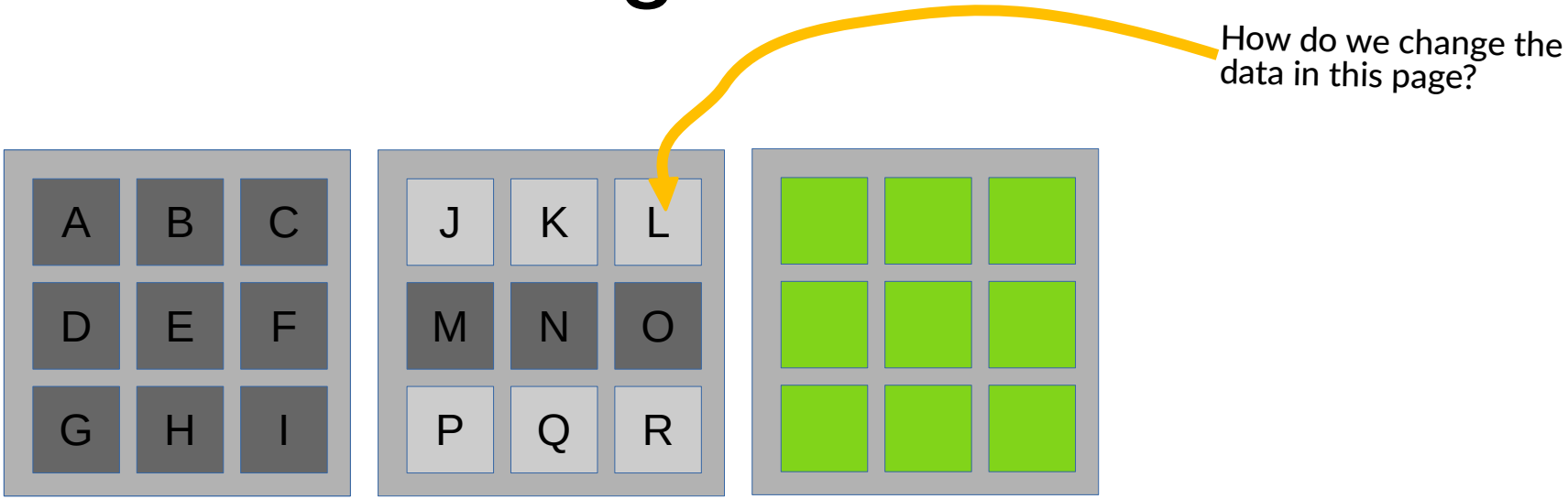
GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Garbage Collection

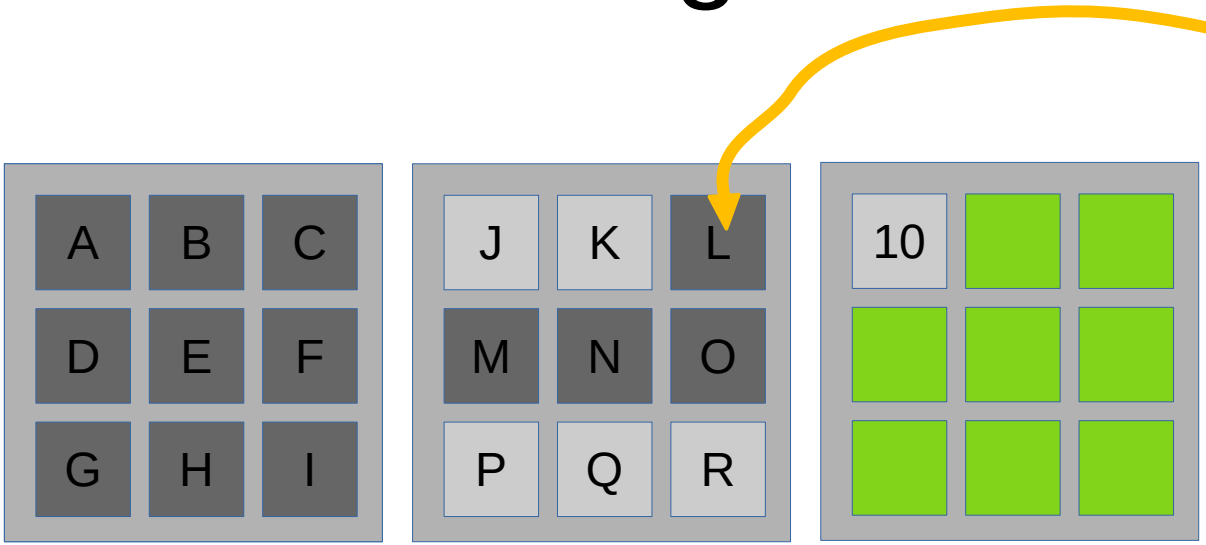


GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Garbage Collection



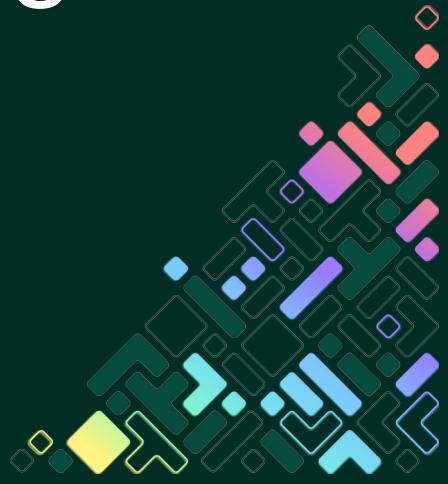
Garbage Collection



- Host wrote 28 pages
- SD Controller wrote 37 pages
- WAF: 1.32x

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Where does it go wrong?



Leaking Electrons and Oxide Wear

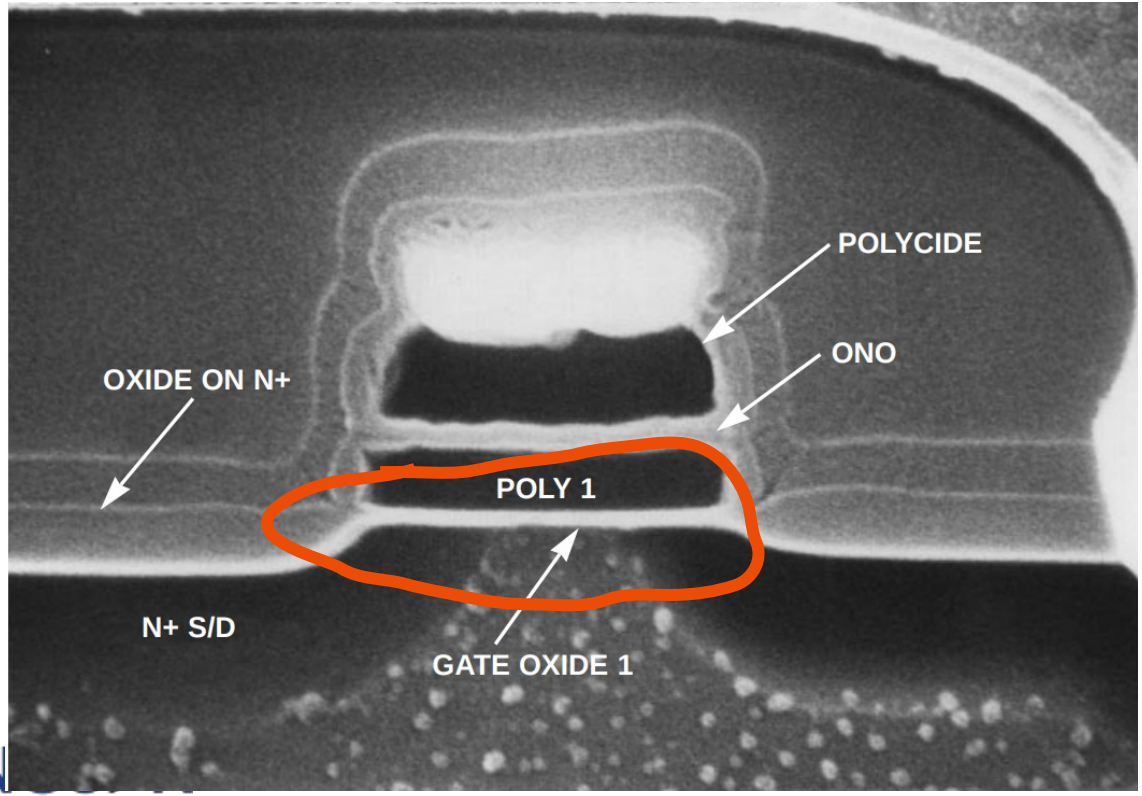
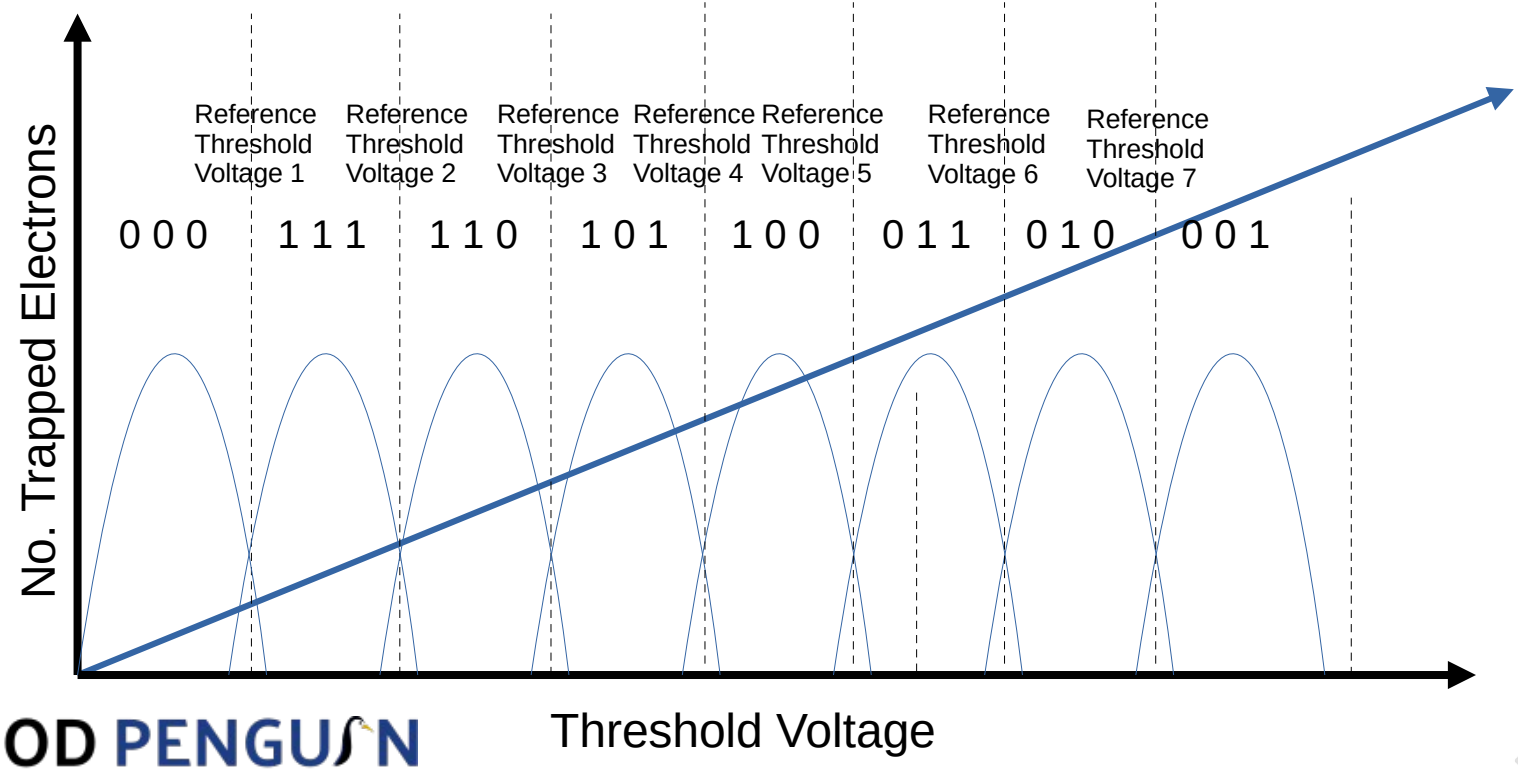


Photo by ICE, “Memory 1997” 22482 (see https://dokumen.tips/download/link/section-10-flash-memory-technology.html)

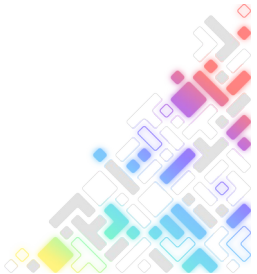
TLC



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

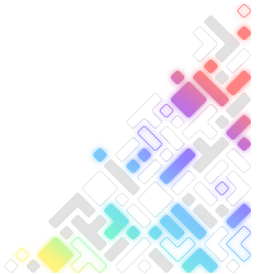
Limited Lifetime

- SLC: 10,000 P/E
- MLC: 3,000 P/E
- TLC: 1,000 P/E
- Wear Levelling helps even wear



Read / Program Disturb

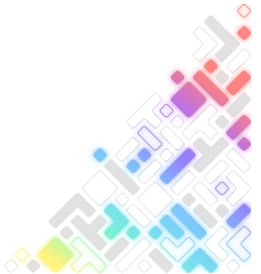
- Reading a NAND cell can increase threshold voltage on related cells
- Due to parasitic capacitive coupling programming one cell can adjust threshold voltage on nearby cells
- ECC can mitigate these effects



ECC

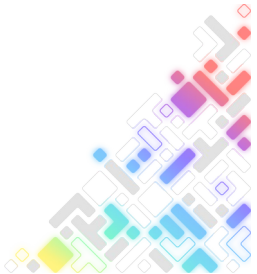
DATA INTEGRITY

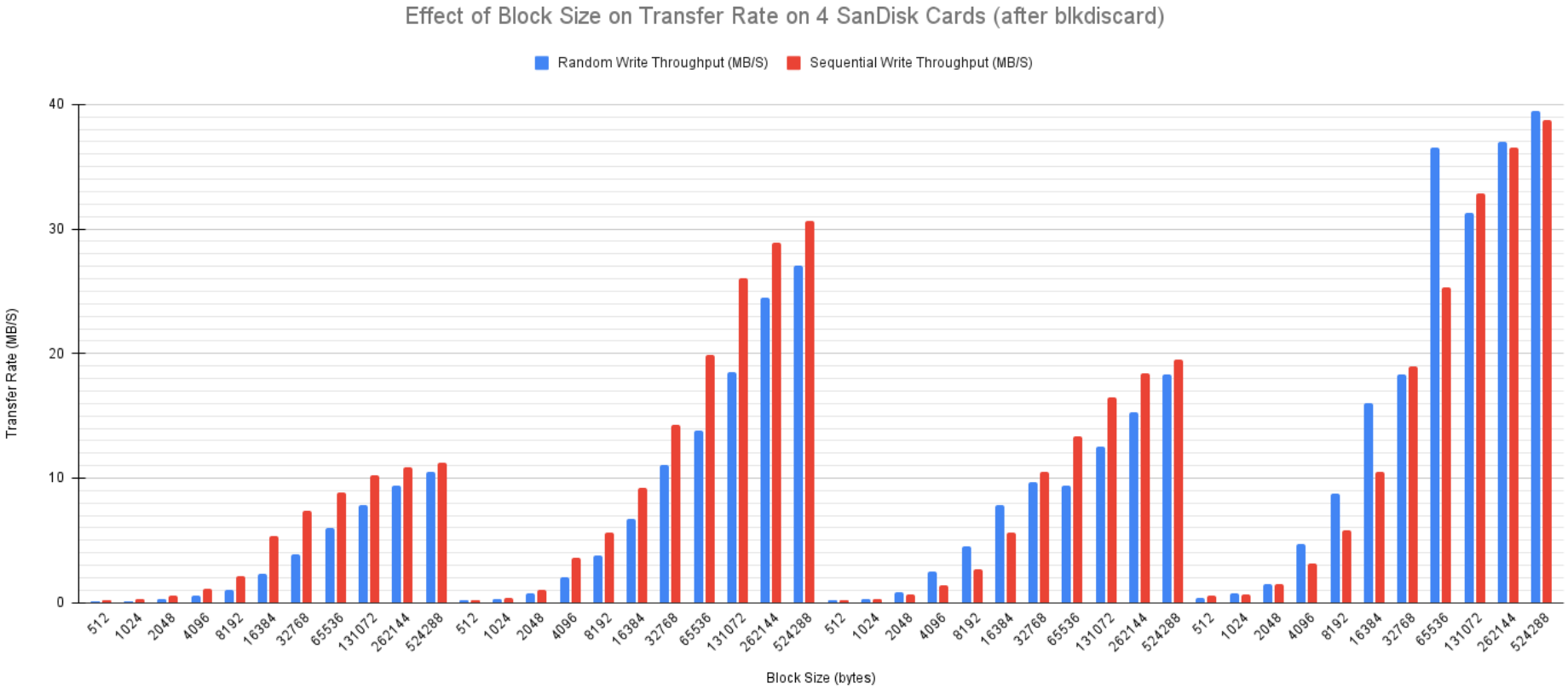
- 100,000 Program/Erase cycles (with 1bit/512byte ECC)



Firmware

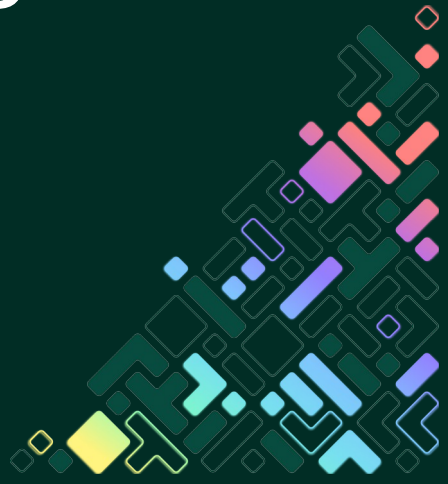
- Goal of firmware is to overcome these challenges through:
  - ECC, Wear Levelling, Garbage Collection, FTL
- Yet completely black box
- Aim is to present uniform block interface





GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

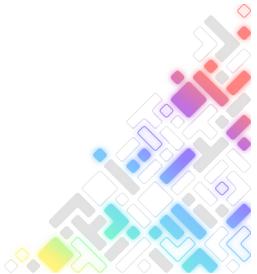
Experimental Testing

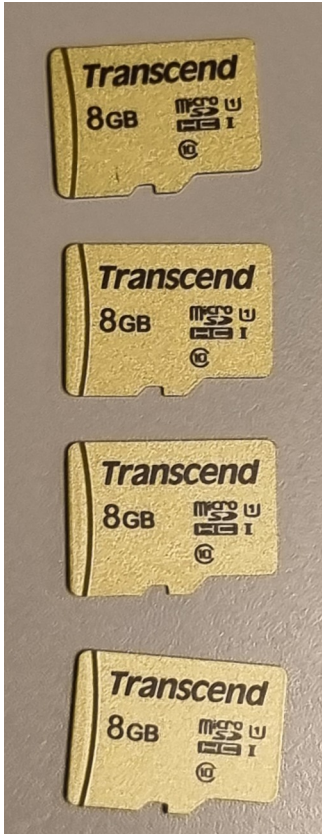


GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

SD Card Test

- How long will they take to break?
- What effect does access patterns/size have?
- How will they break?



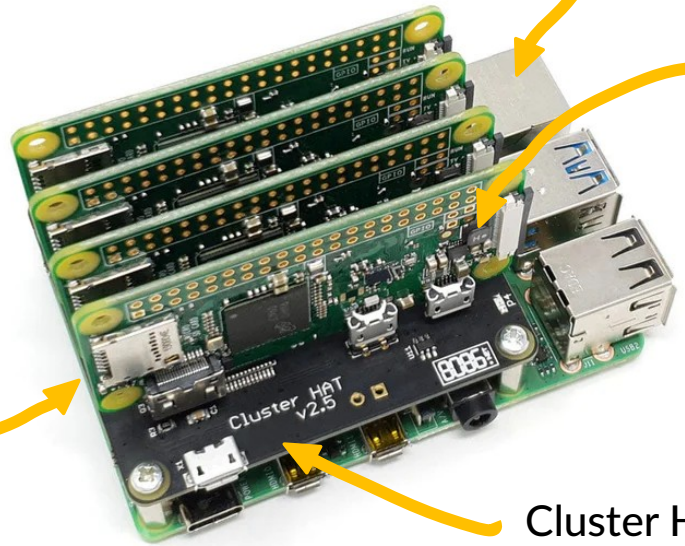


GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Test Setup

Raspberry Pi 4

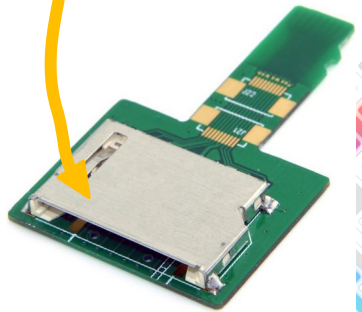
4x Pico Zero 2 W



4x Micro SD Cards

Cluster Hat

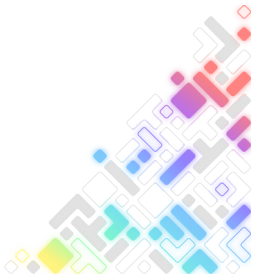
MicroSD to SD



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

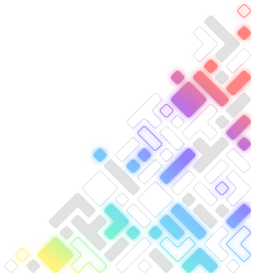
Test Setup

- Use fio (Flexible I/O Tester) to write full SD card and verify its contents:
  - 512 Kb Sequential
  - 512 Kb Random, 128 Kb Random, 4 Kb Random
- Capture logs (kernel/fio) and block stats



Estimate Life

- 7.28 GB SD Card
- MLC with 3000 P/E cycles
- 7.28GB x 3000 = 21.3 TBW ?



Estimate Life

microSD Cards
USD500S

Insertion/Removal Cycles	10,000
Note	Speed may vary due to host hardware, software, usage, and storage capacity.

microSD Cards
USDC10V

Endurance (Max.)	32 GB: 6,000 hours 16 GB: 3,000 hours
Insertion/Removal Cycles	10,000
Note	Speed may vary due to host hardware, software, usage, and storage capacity. Endurance is based on Full HD (1920x1080P) video content recorded at 26 Mbps on Transcend DrivePro series. Before purchasing the High Endurance microSD card, please check the maximum capacity of memory card supported by your dashcam.

microSD Cards
microSDHC/SDXC420T

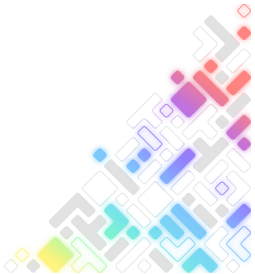
Terabytes Written (TBW)	Up to 640 TBW
Note	• Speed may vary due to host hardware, software, usage, and storage capacity. • Terabytes Written (TBW) expresses the endurance under the highest capacity.

- Endurance: 3K P/E cycles (Program/Erase cycles) guaranteed

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

As of April 2024, see datasheets at https://www.transcend-info.com/Embedded/Product/memory-cards

...and then we waited….



Andy 8:55 AM
First SD card failure spotted on 2GB cluster (512kb random write test) in the early hours of yesterday morning. It's weird because subsequent runs pass.

in my head i expected the cards to fail in such a way where it transitions from all good, to read-only entire sd card, i guess it's not like that

...and then we waited….

Marc 3:46 PM
It'll die 5 mins after your presentation...
1

Philip Boucherat 2 months ago
Are they lasting longer than expected then?

1 reply

Also sent to the channel

Andy 2 months ago
No, I think my expectation has been realigned with reality.

If you take a 32GB card, and assume it has MLC flash, well MLC flash has between 3k and 10k P/E cycles before it **may** go bad (on average).

So you'd need to write at least 3k x 32GB to the SD card before you

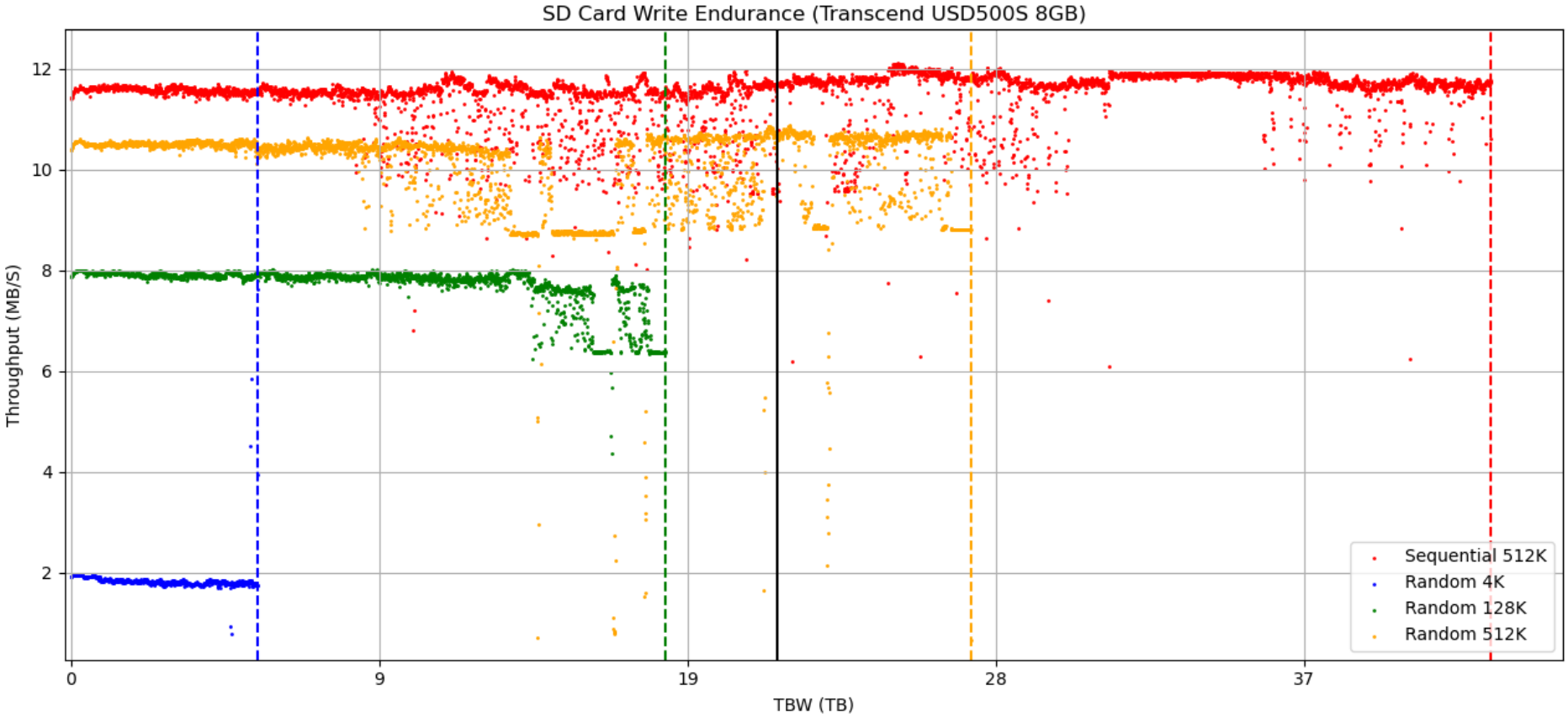
Andy 1:52 PM
Still going 🙂 It must have reached more than 100 TBW now for t

I've been regretting the fact that we don't read the same amount o something does a read and it gets an uncorrectable ECC failure. Ho should be ok, but that does make assumptions about how the SD c a slightly different test on them (i'll still keep the 4 sd cards that sta later) (edited)

Andy 5:19 PM
Note to future self, when performing SD cards make sure the data you write i

The first batch of tests we got running on the 64gb card has a 'testing' phase back. All good, and that test is still running.

For the later batches with the smaller SD cards, I modified the tests for the re back, then repeat. I 'assumed' the data written would be random, and it is, exc fio needs to be set to 0 (https://fio.readthedocs.io/en/latest/fio_doc.html#cm everytime you start fio. So potentially the SD cards in the 2GB card case have

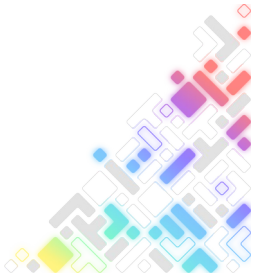


GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

How did it fail?

```
k][w=7 IOPS][eta 01h:22m:06s]^MJobs: 1 (f=1): [w(1)][25.1%][w=2255k][w=4 IOPS][eta 01h:22m:02s]^MJobs: 1 (f=
][25.1%][w=3065k][w=5 IOPS][eta 01h:22m:04s]^MJobs: 1 (f=1): [w(1)][25.1%][w=2043k][w=3 IOPS][eta 01h:22m:04
Jobs: 1 (f=1): [w(1)][25.1%][w=2043k][w=3 IOPS][eta 01h:22m:03s]^MJobs: 1 (f=1): [V(1)][25.2%][r=2043k,w=255
=14 IOPS][eta 01h:21m:48s]           ^M<p4> crc32c: verify failed at file /dev/mmcblk0 offset 2041577472, le
<p4>        Expected CRC: d4b5eea9
<p4>        Received CRC: 4b21a5a9
<p4> fio: pid=10978, err=84/file:io_u.c:2132, func=io_u_queued_complete, error=Invalid or incomplete multiby
<p4>
client <172.19.180.4>: exited with error 1
device: (groupid=0, jobs=1): err=84 (file:io_u.c:2132, func=io_u_queued_complete, error=Invalid or incomplet
  write: IOPS=2, BW=1238KiB/s (1267kB/s)(2000MiB/1654694msec); 0 zone resets
```

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS



How did it fail?

```
Mar 26 13:30:26 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 26 13:30:39 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 26 13:30:59 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 26 13:31:19 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:40:34 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:40:44 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:40:56 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:41:06 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:41:17 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:41:36 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:41:56 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:42:07 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:42:17 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:42:37 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:42:48 p1 kernel: mmc0: timeout waiting for hardware interrupt.
Mar 30 11:42:59 p1 kernel: mmc0: timeout waiting for hardware interrupt.
```

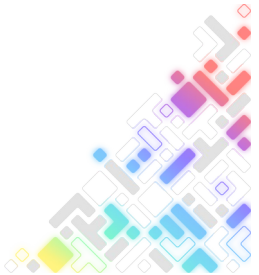
```
^MJobs: 1 (f=1): [w(1)][4.0%][eta 19h:37m:08s]^M<p3> fio: io_u error on file /dev/mmcblk0: Input/output error: write offset=6828457984, buflen=131072
<p3> fio: pid=16326, err=5/file:io_u.c:1834, func=io_u error, error=Input/output error
<p3>
client <172.19.180.3>: exited with error 1
device: (groupid=0, jobs=1): err= 5 (file:io_u.c:1834, func=io_u error, error=Input/output error): pid=16326: Mon Mar 11 19:47:39 2024
  write: IOPS=0, BW=122KiB/s (124kB/s)(354MiB/2980237msec); 0 zone resets
```

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

How did it fail?

```
Mar 24 00:00:23 p2 kernel: [5228276.609021] I/O error, dev mmcblk0, sector 1138688 op 0x1:(WRITE) flags 0x8800 phys_seg 1 prio class 2
Mar 24 00:00:32 p2 kernel: [5228285.665283] mmc0: previous command never completed.
Mar 24 00:00:32 p2 kernel: [5228285.768906] mmc0: previous command never completed.
Mar 24 00:00:32 p2 kernel: [5228285.870455] mmc0: previous command never completed.
Mar 24 00:00:32 p2 kernel: [5228285.976525] mmc0: previous command never completed.
Mar 24 00:00:32 p2 kernel: [5228286.080881] mmc0: previous command never completed.
Mar 24 00:00:32 p2 kernel: [5228286.184418] mmc0: previous command never completed.
Mar 24 00:00:32 p2 kernel: [5228286.296146] mmc0: previous command never completed.
Mar 24 00:00:32 p2 kernel: [5228286.401797] mmc0: previous command never completed.
Mar 24 00:00:32 p2 kernel: [5228286.504329] mmc0: previous command never completed.
Mar 24 00:00:33 p2 kernel: [5228286.606558] mmc0: previous command never completed.
Mar 24 00:00:33 p2 kernel: [5228286.708363] mmc0: previous command never completed.
Mar 24 00:00:33 p2 kernel: [5228286.816803] mmc0: previous command never completed.
Mar 24 00:00:33 p2 kernel: [5228286.921065] mmc0: previous command never completed.
Mar 24 00:00:33 p2 kernel: [5228286.925551] mmcblk0: recovery failed!
Mar 24 00:00:33 p2 kernel: [5228286.925598] I/O error, dev mmcblk0, sector 1138688 op 0x1:(WRITE) flags 0x8800 phys_seg 1 prio class 2
Mar 24 00:00:39 p2 kernel: [5228293.018673] mmc0: previous command never completed.
Mar 24 00:00:39 p2 kernel: [5228293.120985] mmc0: previous command never completed.
Mar 24 00:00:39 p2 kernel: [5228293.223807] mmc0: previous command never completed.
Mar 24 00:00:39 p2 kernel: [5228293.329637] mmc0: previous command never completed.
Mar 24 00:00:39 p2 kernel: [5228293.435297] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228293.541071] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228293.645332] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228293.756562] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228293.860378] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228293.962402] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228294.064383] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228294.170404] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228294.276697] mmc0: previous command never completed.
Mar 24 00:00:40 p2 kernel: [5228294.282890] mmcblk0: recovery failed!
```

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS



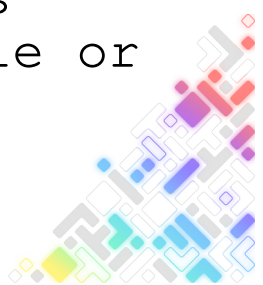
How did it fail?

```
Mar 26 18:17:01 p4 kernel: mmc1: new high speed SDIO card at address 0001
Mar 26 18:17:01 p4 kernel: mmc0: card never left busy state
Mar 26 18:17:01 p4 kernel: mmc0: error -110 whilst initialising SD card
Mar 26 18:17:02 p4 kernel: Freeing initrd memory: 6164K
Mar 26 18:17:02 p4 kernel: mmc0: card never left busy state
Mar 26 18:17:02 p4 kernel: mmc0: error -110 whilst initialising SD card
Mar 26 18:17:02 p4 kernel: mmc0: card never left busy state
Mar 26 18:17:02 p4 kernel: mmc0: error -110 whilst initialising SD card
Mar 26 18:17:02 p4 kernel: mmc0: card never left busy state
Mar 26 18:17:02 p4 kernel: mmc0: error -110 whilst initialising SD card
Mar 26 18:17:02 p4 kernel: Freeing unused kernel image (initmem) memory: 432K
```

```
[   23.536543] mmc0: SDHBLC 0x00000008
[   23.536554] mmc0: ============================================
[   24.886438] mmc0: card never left busy state
[   24.890973] mmc0: tried to HW reset card, got error -110
[   24.901810] mmcblk0: recovery failed!
[   24.905652] I/O error, dev mmcblk0, sector 0 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 2
[   24.914436] Buffer I/O error on dev mmcblk0, logical block 0, async page read
[   24.932817] mmcblk0: recovery failed!
[   24.936666] I/O error, dev mmcblk0, sector 0 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 2
[   24.945451] Buffer I/O error on dev mmcblk0, logical block 0, async page read
[   24.963827] mmcblk0: recovery failed!
[   24.967717] I/O error, dev mmcblk0, sector 0 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 2
[   24.976461] Buffer I/O error on dev mmcblk0, logical block 0, async page read
[   24.983930]  mmcblk0: unable to read partition table
[   24.984117] mmcblk0: mmc0:59b4 USDU1 7.28 GiB (ro)
[   36.413169] Freeing unused kernel image (initmem) memory: 432K
```

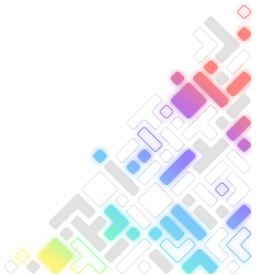
How others fail (Sandisk)

```
$ # Write to card
$ echo world > /media/andy/0CCD-CCEB/hello
$ cat /media/andy/0CCD-CCEB/hello world
$ sudo umount /media/andy/0CCD-CCEB


$ # Read file after remounting
$ cat /media/andy/0CCD-CCEB/hello

$ # Try again after reinsert or dropping caches
$ cat: /media/andy/0CCD-CCEB/hello: No such file or
directory
```

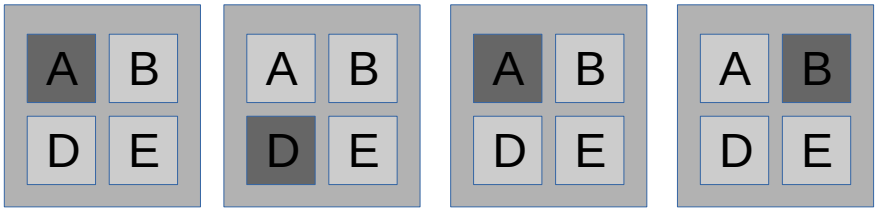



Estimate Life Revisited

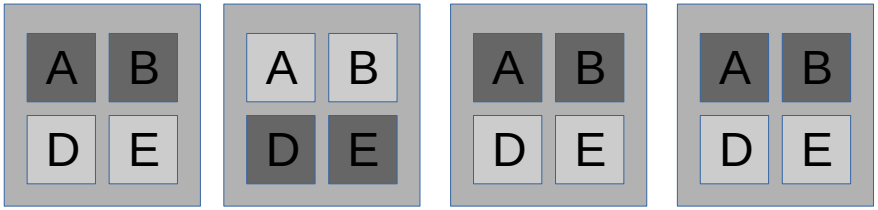
- We achieved beyond 42.87 TBW
- At least 6030 P/E cycles
- Write amplification:
  - 4kb Random: 5.64 TBW (> 7.60x)
  - 128kb Random: 17.98 TBW (> 2.38x)
  - 512kb Random: 27.17 TBW (> 1.57x)



Write Amplification



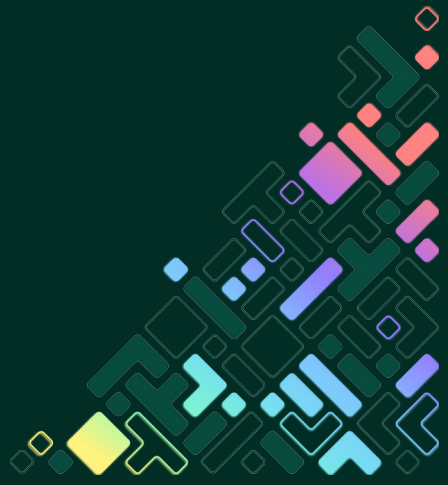
4 erases to provide 1 new block (4x)



4 erases to provide 2 new blocks (2x)

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Practical Steps



Reduce workload

- Reducing workload will reduce P/E wear and prolong life
- Determine volume of I/O with block layer statistics

```
root@p1:~# cat /sys/block/mmcblk0/stat
30312929        0 685219992 684880008 30017800        0 673456128 1288243661        1 2572674500 1973123667        0
root@p1:~# cat /proc/diskstats | grep mmcblk0
 179       0 mmcblk0 30313575 0 685881496 684894717 30018420 0 674091008 1288269702 1 2572727760 1973164419 0 0 0 0 0
```

Write sectors (512b)

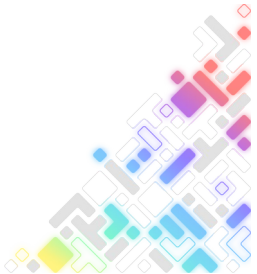
Write I/O requests

- Identify and remove unnecessary writes (blktrace)
- R/O filesystems, access times, logs, temporary data

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

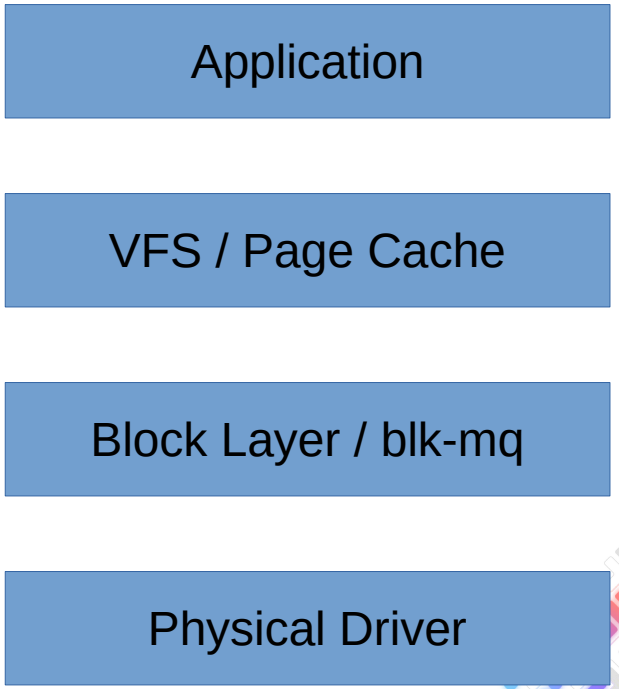
Reduce amplification

- Favour sequential large writes with limit amplification and improve performance
- Determine average write size of I/O with block layer statistics or iostat
- This is really hard...



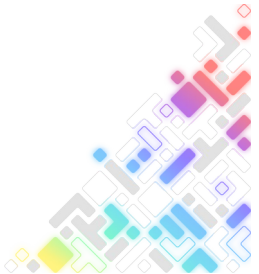
Path through kernel

- Application writes to filesystem
- Data sits in page cache
- The kernel decides when and how these dirty pages are written out
- Block layer / IO schedule can reorder, combine, split write requests
- Application workload will look different to workload at device



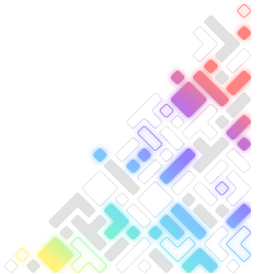
Reduce amplification

- Consider filesystem type and cluster size (how card formatted)
- Experiment with IO schedulers and tuneables
    - mq-deadline and large nr_requests
- Watch out for direct IO and over zealous use of sync
- Use fio to replay a blktrace workload (quickly) and iterate
- Use discard, fstrim, blkdiscard



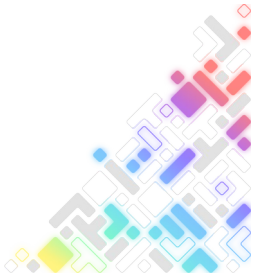
Over provision

- SD card life varies with amount of NAND
- Thanks to wear levelling, with more NAND the card will last longer
- Buy a larger card or use a card with a higher TBW
- Unused NAND reduces write amplification



Speak to Vendor

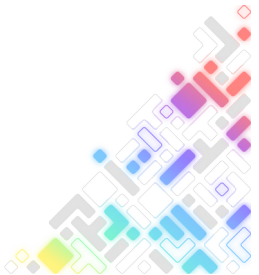
- The firmware can make a large difference in terms of amplification
- The firmware can provide additional commands that indicate wear
- There are a wide range of cards to specific purposes
- Maybe SD cards are not the right solution for you



Performance

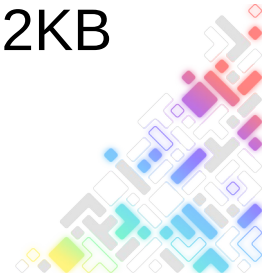
- Is the SD card a bottleneck? Use iostat:

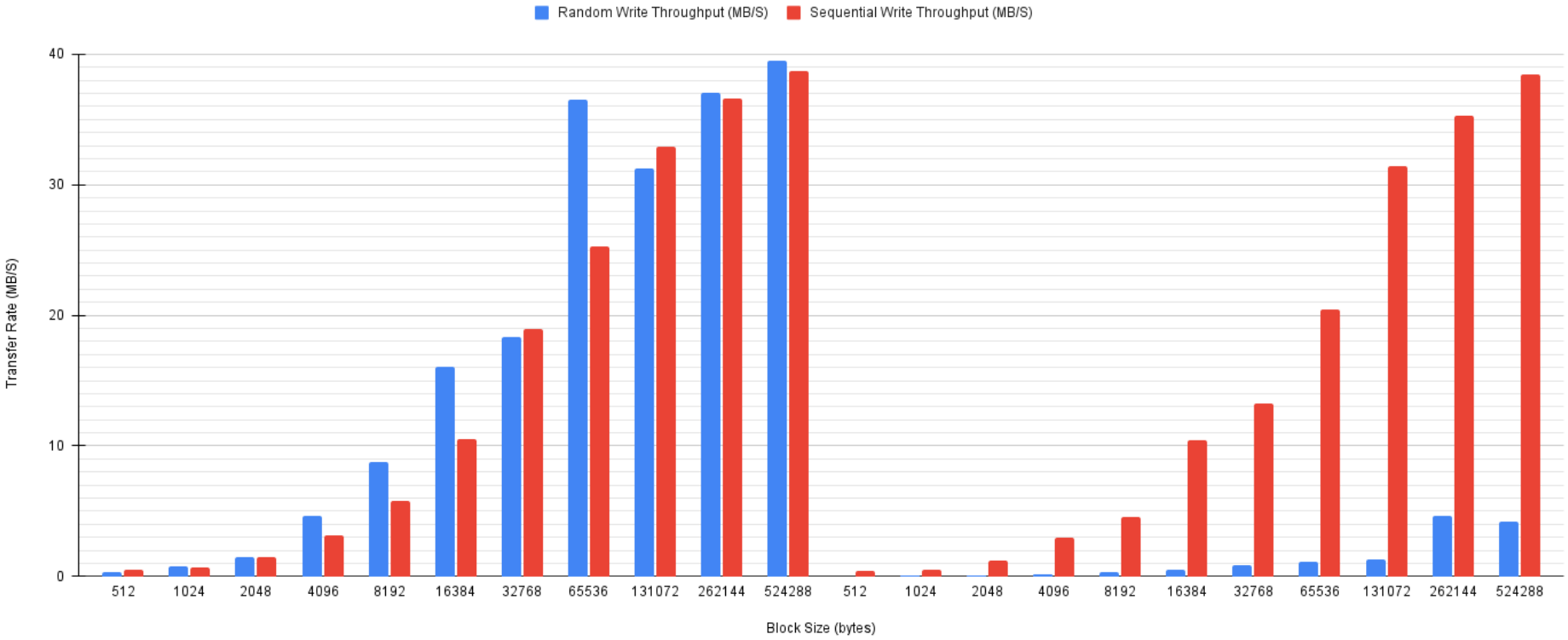
  - Utilisation: %util
  - Average Write Size: wareq-sz
  - Average Throughput: wkB/s
  - IOWait: %iowait



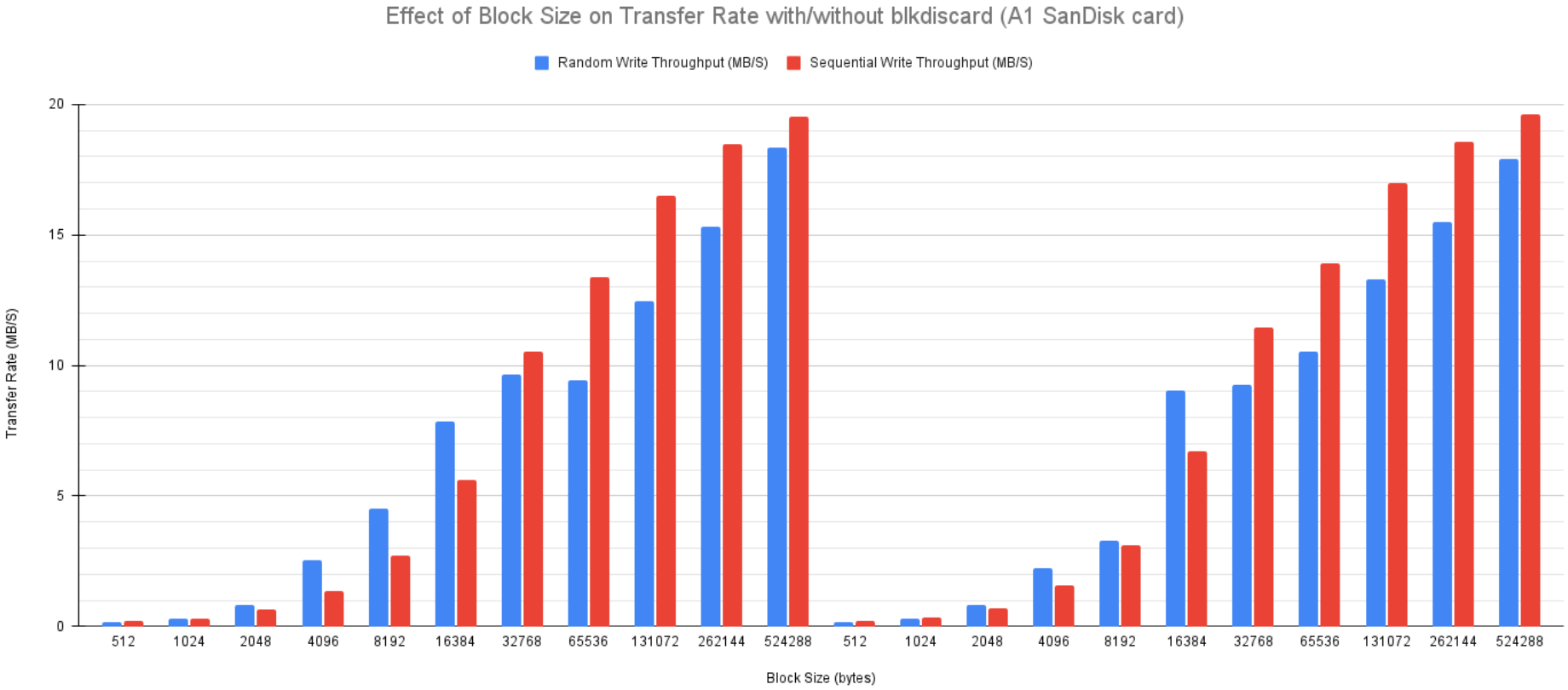
Performance

- Many speed classes make very specific assumptions not achieved on Linux (Particularity for video classes and UHS)
- Other assumptions assume use of exFAT, write size, etc
- A1 cards guarantee IOPS with 4KB RU
- SD card specification assumes Access Unit (erase block group size) is 4MB (cards up to 32GB) and 64MB (larger cards)
- SD card specification assumes Read Unit size is likely 512KB





GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS



GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

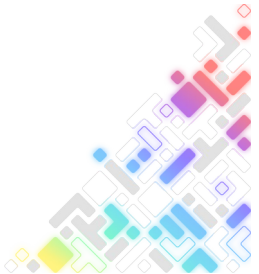
KrillKounter

- KrillKounter persistently logs block layer statistics
- Really simple, but allows you to determine wear on card
- We hope in future it will be expanded to provide predictions and support for other flash media
- github.com/The-Good-Penguin/tgp-krill-kounter

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS

Summary

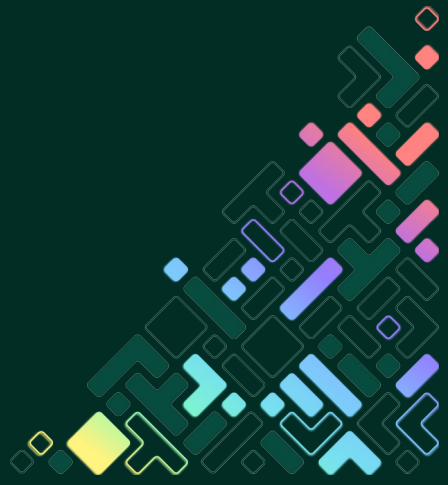
- Access patterns have an impact on performance and wear
- Sequential large accesses are usually better
- Over provision for improved endurance
- Don’t rely on SD card performance classes
- Measure at the block layer and optimise
- Choose an SD card wisely



Questions?

amurray@thegoodpenguin.co.uk
www.thegoodpenguin.co.uk

GOOD PENGUIN
DEPENDABLE SOFTWARE SOLUTIONS



EMBEDDED LINUX CONFERENCE